Pedro Javier García García

DISEÑO DE UNA TÉCNICA ESCALABLE Y EFICIENTE PARA EL CONTROL DE LA CONGESTIÓN EN REDES DE INTERCONEXIÓN

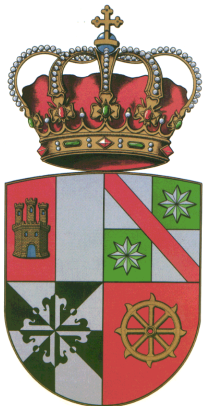
I.S.B.N. Ediciones de la UCLM
978-84-8427-602-9

Ediciones de la Universidad
de Castilla-La Mancha

Cuenca, 2008

UNIVERSIDAD DE CASTILLA-LA MANCHA

DEPARTAMENTO DE SISTEMAS INFORMÁTICOS



Diseño de una Técnica Escalable y Eficiente para el Control de Congestión en Redes de Interconexión

Tesis Doctoral
presentada al Departamento de Sistemas Informáticos
de la Universidad de Castilla-La Mancha
para la obtención del título de
Doctor en Informática

Presentada por:
Pedro Javier García García

Dirigida por:
Dr. D. José Duato Marín
Dr. D. Francisco José Quiles Flor

Albacete, Noviembre de 2006

A Rocío, mi mujer,
y a Rocío y Pedro, nuestros dos tesoros

Agradecimientos

"Vivir es más que un derecho,
es el deber de no claudicar"
de "Libertad", L.E. Aute.

Salvo en casos excepcionales, el trabajo del investigador científico no es un camino de rosas. Supongo que mi caso no es una excepción, ya que no puedo decir que el camino seguido hasta terminar la tesis doctoral que se presenta en esta memoria haya estado exento de dificultades. Sin embargo, en todo momento he contado con el apoyo, el afecto y la colaboración de numerosas personas, y esto ha sido fundamental de cara a superar todos los inconvenientes que han surgido durante la realización de la tesis. A todas estas personas me gustaría agradecer, en los siguientes párrafos, el haberme acompañado, de una forma u otra, a lo largo de este camino.

Sin duda, debo nombrar en primer lugar a mi esposa y a nuestros hijos. Todos ellos han sufrido más que nadie mis ausencias, mis presencias con la mente en otra parte, mi mal humor, mi cansancio, mi nerviosismo, y tantos otros "efectos colaterales" de la realización de la tesis. Además de agradecerles la paciencia y comprensión que han mostrado conmigo, y su cariño, que jamás me ha faltado, sólo puedo decirles que espero poder compensar los malos momentos pasados dedicándoles más tiempo de ahora en adelante. Espero igualmente que mis hijos entiendan algún día que, si su padre no les dedicó todo el tiempo que merecían, fue por hacer algo que creía que merecía la pena.

También tengo que agradecer a mis padres todo el cariño y el apoyo incondicional que he recibido de ellos, no sólo durante la realización de la tesis, sino durante toda mi vida. Además, he de agradecerles los conocimientos y valores que ellos me han enseñado desde pequeño, bien expresamente, bien mediante su ejemplo constante de bondad, abnegación, sacrificio y espíritu de superación ante la adversidad. Igualmente, jamás me ha fallado el cariño y el ánimo de mi hermano, que, a pesar de ser más joven, también me ha enseñado varias e importantes lecciones vitales, sobre todo en lo que a constancia y voluntad se refiere.

Evidentemente, no pueden faltar en estos agradecimientos mis directores de tesis, José Duato y Francisco Quiles. La valía de ambos como investigadores es de sobra conocida, nacional e internacionalmente, por lo que incidir por mi parte en este aspecto resultaría casi redundante. Baste en este sentido decir que cualquier doctorando se consideraría afortunado por tener dos directores de semejante prestigio y experiencia. Además de todo lo que he aprendido de ellos, en lo profesional y en lo personal, he

de agradecerles también la confianza depositada en mí para desarrollar la línea de investigación de la que es fruto esta tesis.

También debo agradecer especialmente toda la ayuda, el apoyo y las ideas recibidos de José Flich, de quien también he aprendido bastante en este tiempo. Su constante soporte y colaboración han resultado esenciales para la realización de esta tesis, de tal modo que espero poder compensar de alguna manera en el futuro todo el tiempo y el esfuerzo que ha dedicado a esta línea de investigación. No en vano hemos compartido innumerables horas de trabajo, que sin duda se han hecho más llevaderas gracias a su capacidad investigadora y a su calidad humana, y también gracias a la camaradería que se ha establecido entre nosotros. Además, tengo que agradecer su total disposición para atenderme en cualquier momento, y, a la vez, alabar la paciencia que ha mostrado conmigo en infinidad de ocasiones.

Cabe mencionar también la ayuda y la información ofrecidas por Ian Johnson y Finbar Naven, de Xyratex, que han aportado siempre a la investigación desarrollada en esta tesis un enfoque eminentemente realista y práctico, fundamental para la obtención de los resultados finales.

Me gustaría destacar también el apoyo y la ayuda que me han brindado en todo momento mis compañeros del grupo de investigación en Redes y Arquitecturas de Altas Prestaciones (RAAP). Me considero afortunado por pertenecer a un grupo de investigación que, gracias al esfuerzo de todos sus miembros, ha conseguido consolidarse plenamente, siendo valorado y respetado a nivel nacional e internacional. Sin duda, mi formación como investigador y como persona no hubiera sido la misma sin la experiencia adquirida en el grupo y sin los sabios consejos de algunos de sus miembros. Además, tanto con ellos como con el resto de compañeros del Departamento de Sistemas Informáticos, he compartido tal cantidad de vivencias a lo largo de los últimos años, que en muchos casos nuestra relación puede considerarse sin duda de amistad.

No puedo olvidar tampoco agradecer la amabilidad con la que he sido recibido en aquellos grupos donde he realizado estancias de investigación durante el doctorado, concretamente el Grupo de Arquitecturas Paralelas, en la Universidad Politécnica de Valencia, y el *Computer Architecture Group*, en la Universidad de Mannheim. He de decir que jamás, durante el tiempo que he pasado trabajando con estos grupos, me he sentido un extraño, ya que ambos me han acogido y tratado como uno más de sus miembros, ofreciéndome toda la ayuda posible. Además, creo haberme enriquecido muchísimo, como investigador y como persona, gracias a estas estancias, y creo también haber encontrado en ellas amistades auténticas.

Por último, me gustaría agradecer los ánimos y el afecto que siempre me han brindado todas aquellas personas que han confiado en mí en todo momento, interesándose siempre por la evolución de la tesis. Entre ellos me gustaría destacar a Carmelo Garrido, a mis tíos y primos, y a mi panda de amigos "impresentables".

A todos los mencionados, de nuevo, muchas gracias.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Contexto

A lo largo de las últimas décadas, y especialmente en la más inmediata, la evolución en el campo de las redes de interconexión ha sido constante. Durante estos años, los avances tecnológicos han permitido aumentar el ancho de banda de los enlaces, dotar de mayor flexibilidad a la topología e incrementar la eficiencia tanto de las técnicas de conmutación y control de flujo como de los algoritmos de encaminamiento, lo que ha supuesto en conjunto una muy significativa mejora de las prestaciones de las redes que ha tenido una enorme repercusión en el mundo de la computación y la comunicación. Así, hoy en día podemos encontrar redes de interconexión de altas prestaciones que son empleadas no sólo en los clásicos computadores masivamente paralelos (multicomputadores y multiprocesadores), sino también en otros sistemas, como redes de área local (LANs), redes de área de sistema (SANs), *clusters* de PCs y estaciones de trabajo, o conmutadores IP. Más aún, la popularidad e implantación de estos sistemas crece continuamente, traspasando los límites del ámbito estrictamente científico. A ello han contribuido dos factores fundamentales:

- La aparición en el mercado de redes de interconexión comerciales de altas prestaciones (*Myrinet, Servernet, Autonet, Infiniband, Quadrics,* etc.), que permiten, a un coste más o menos asequible, construir sistemas con gran capacidad de cómputo y con alta velocidad en la transmisión de datos.
- La creciente demanda de aplicaciones y servicios con grandes requisitos de cómputo y comunicación, a los que los mencionados sistemas pueden dar soporte.

Esta proliferación de sistemas basados en redes de interconexión de altas prestaciones no ha hecho sino aumentar el interés de los investigadores en desarrollar técnicas que permitan a estas redes ofrecer prestaciones cada vez mejores. Se cierra de este

modo el clásico círculo en el que los avances en investigación y desarrollo mejoran ciertos sistemas y servicios, lo que genera una demanda que a su vez impulsa un mayor esfuerzo investigador y de desarrollo.

En el caso de las redes de interconexión, este esfuerzo se ha plasmado no sólo en el desarrollo de nuevas técnicas, sino también en el replanteamiento y adaptación de otras técnicas y soluciones ya existentes. Efectivamente, en una situación de evolución constante de la tecnología, sujeta además a las exigencias del mercado, algunas técnicas y soluciones que han sido indiscutiblemente válidas en un momento dado, o para un entorno dado, han dejado de serlo en otro momento, o para otro entorno. En consecuencia, ciertos problemas, que se creían solucionados definitivamente, han vuelto a surgir al cambiar las condiciones asumidas en su solución.

En concreto, la congestión en redes de interconexión se consideraba hasta hace muy poco un problema resuelto en sistemas reales gracias a la habitual práctica de sobredimensionar la red, empleando un número de componentes (enlaces, conmutadores) mucho mayor del estrictamente necesario para interconectar los terminales. Esto garantiza una muy baja utilización de los enlaces, lo que minimiza a su vez la contención de los paquetes por conseguir los mismos recursos de la red (puertos o enlaces de salida), garantizándose así la virtual ausencia de congestión.

Para valorar la importancia de esta garantía, hay que tener en cuenta que la existencia de congestión suele afectar a las prestaciones ofrecidas por la red, que, en estas circunstancias, pueden degradarse drásticamente. La causa concreta de esta degradación es un conocido fenómeno, asociado a la congestión, denominado "bloqueo de cabeza de línea" (o *Head-Of-Line blocking*). Este fenómeno se produce cuando paquetes almacenados en una cola no pueden avanzar, incluso estando libres los recursos que solicitan, al estar temporalmente bloqueado el paquete situado en cabeza de esa cola. Las consecuencias de estas situaciones son más graves en aquellas redes donde no se permite el descarte de paquetes (redes sin pérdidas), como es precisamente el caso de la mayoría de las actuales redes de interconexión de altas prestaciones. Además, debido al control de flujo, la congestión se propaga rápidamente por la red, formándose los llamados "árboles de congestión".

Ahora bien, la "tradicional" y sencilla solución al problema de la congestión mediante el sobredimensionamiento de la red ha acabado quedando obsoleta debido a dos importantes inconvenientes:

- El aumento del coste de los componentes de la red. Hoy en día, los enlaces y conmutadores de redes comerciales de altas prestaciones resultan caros en comparación con los procesadores empleados en los sistemas actuales.
- El mayor consumo de potencia de los componentes de las modernas redes de altas prestaciones. Esto es consecuencia, fundamentalmente, del aumento de la

velocidad de los enlaces, unido al hecho de que el consumo de potencia en los enlaces actuales es prácticamente independiente de su grado de utilización para la transmisión de tráfico convencional, debido al incremento de la señalización y mensajería de control.

Aunque existen propuestas de técnicas de escalado en voltaje y frecuencia orientadas a reducir el consumo de potencia en redes de interconexión, hasta ahora no han demostrado ser muy eficientes. Y, en cualquier caso, estas técnicas sólo resolverían el segundo de los inconvenientes anteriormente expuestos, dejando sin resolver el problema del coste de los componentes.

En conclusión, no parece viable, en las actuales circunstancias, construir redes de interconexión sobredimensionadas para sistemas reales. Desde el punto de vista de las posibilidades de conexión de las tecnologías de red actuales, no supone ningún problema construir redes más reducidas, pues la gran mayoría permiten la conexión de múltiples terminales a los conmutadores. Ahora bien, esto implica que existirá la posibilidad de congestión en la red, al producirse una mayor contención por los más escasos recursos disponibles en la red. Por tanto, la congestión en redes de interconexión ha vuelto a plantearse como un problema que requiere la aplicación de otras soluciones. Es necesario actualmente, en definitiva, emplear técnicas específicas de control de la congestión en las redes de interconexión.

Los trabajos que se describen en la presente memoria pretenden, precisamente, responder a esta necesidad, proponiendo una técnica de control de congestión que ofrezca un mejor funcionamiento que las propuestas anteriormente.

1.2. Antecedentes

El control de congestión en redes de interconexión ha sido objeto de estudio durante décadas. Frente a la simple solución de la congestión mediante el descarte de paquetes, que sólo puede emplearse en redes que admiten pérdidas (como es el caso de Internet), las soluciones para redes sin pérdidas se basan en técnicas elaboradas que, a grandes rasgos, pueden clasificarse en tres grandes grupos:

- **Técnicas de evitación** (o proactivas). Requieren una planificación previa de la asignación de recursos para garantizar la ausencia de congestión. En general, estas técnicas se han propuesto para entornos con requisitos de garantía de calidad de servicio (*Quality of Service, QoS*).
- **Técnicas de prevención**. Se basan en controlar el tráfico durante el funcionamiento de la red, de tal modo que la congestión no pueda aparecer. Este control implica la toma de decisiones "en caliente", que pueden suponer limitar (o

modificar) las rutas a seguir por los paquetes, o los accesos a dispositivos de almacenamiento.

- **Técnicas de detección** (o reactivas). En este caso, se permite que la congestión se produzca, pero se dispone de algún tipo de mecanismo encargado de eliminarla cuando se detecta. Esta detección necesita de ciertos indicadores, como pueden ser el número de paquetes almacenados en los *buffers*, o el número de peticiones de acceso a dispositivos de almacenamiento. Una vez detectada, la congestión suele ser notificada a las fuentes que la producen para que reduzcan o cesen su actividad.

Por otra parte, existen otras técnicas que, a diferencia de las encuadradas en la clasificación anterior, no tienen como objetivo garantizar la ausencia o eliminación de la congestión, sino que se centran en eliminar o reducir el *Head-Of-Line (HOL) blocking*. En general, estas técnicas han tenido una amplia difusión, y existen numerosos estudios dedicados a evaluar y mejorar sus prestaciones. Su funcionamiento suele basarse en disponer de *buffers* separados para paquetes con distintos destinos o en proporcionar alguna "vía de escape" que permita a los paquetes avanzar aunque los precedentes en su *buffer* estén bloqueados. Mientras que algunas de estas técnicas atacan el problema del *HOL blocking* a nivel de conmutador, otras lo hacen a nivel de la red completa.

Por último, existen soluciones que retrasan la aparición de la congestión, como pueden ser las técnicas de equilibrado de carga o el uso de encaminamiento adaptativo. Pero en ningún caso estas técnicas permiten evitar la degradación de prestaciones en la red una vez que la congestión aparece.

Aunque las ventajas e inconvenientes de los enfoques expuestos se analizarán con detalle en un capítulo posterior, es necesario en este punto adelantar que, a nuestro juicio, ninguna de las soluciones anteriores satisface plenamente las necesidades actuales de las redes de interconexión en cuanto a control de congestión. Evidentemente, las razones que nos llevan a esta conclusión varían dependiendo de la técnica concreta considerada. Pero, de forma general, cabría destacar los siguientes inconvenientes como los más importantes:

- **Respuesta lenta:** algunos mecanismos destinados a controlar la congestión pueden actuar demasiado tarde en ciertas circunstancias.
- **Eficiencia limitada:** ciertas técnicas no consiguen eliminar totalmente los problemas asociados a la congestión.
- **Alta sensibilidad a las variaciones de tráfico:** la eficiencia de determinadas técnicas depende demasiado del tipo de tráfico presente en la red.
- **Falta de escalabilidad:** algunas técnicas son aplicables en la práctica sólo en redes de tamaño limitado.

Por otra parte, estos inconvenientes no son en absoluto excluyentes entre sí, por lo que ciertas técnicas pueden presentar varios de ellos simultáneamente.

1.3. Motivación

Teniendo en cuenta lo expuesto en los puntos anteriores, nos encontramos en una situación en la que, por una parte, las actuales redes de interconexión de altas prestaciones necesitan emplear alguna técnica de control de congestión, y por otra, las propuestas existentes en este campo bien presentan problemas en su funcionamiento, bien no son aplicables a sistemas reales.

Por todo ello, consideramos que es necesario proponer y desarrollar una nueva técnica de control de congestión que pueda ser implementada en las actuales redes de interconexión de altas prestaciones, y cuya eficiencia sea la máxima, independientemente del tamaño de la red y del tráfico presente en la misma. Dicho de otro modo, consideramos necesaria una nueva técnica de control de congestión que sea a la vez eficiente y escalable.

La obtención de una técnica semejante solucionaría los problemas asociados a la congestión en cualquier circunstancia. Nótese que esto permitiría, entre otras cosas, construir sistemas basados en redes de tamaño reducido sin riesgo a una degradación de prestaciones por la aparición de congestión, lo que a su vez supondría abaratar el coste de la red y reducir significativamente el consumo de potencia.

Además, creemos que una técnica de control de congestión de estas características es perfectamente posible si, por una parte, se analizan y consideran ciertas propiedades del comportamiento del tráfico y, por otra, se aprovechan ciertas facilidades que proporcionan las actuales tecnologías de redes de interconexión. Aun cuando el planteamiento y desarrollo de nuestra técnica será explicado con todo detalle en capítulos posteriores, podemos presentar aquí algunas de estas características del tráfico y la tecnología que nos hacen considerar viable nuestra propuesta:

- En situación de congestión, aquellos flujos de tráfico destinados a puntos congestionados pueden interferir en algunas zonas de la red con los flujos de tráfico destinados a puntos no congestionados. Esta interferencia es la principal causa de aparición del *HOL blocking* y, en consecuencia, de la degradación de prestaciones que puede sufrir la red en estas situaciones. Una técnica capaz de eliminar dicha interferencia acabaría, por tanto, con el principal efecto negativo de la congestión.

- Desde un punto de vista tecnológico, es posible asignar dinámicamente ciertos recursos de la red (*buffers*) para separar flujos congestionados y no congestionados, de manera que no interfieran entre ellos. Esto eliminaría el *HOL blocking*

realmente significativo, que es el introducido por los flujos congestionados en los flujos no congestionados.

- Las conocidas propiedades de localidad espacial y temporal del tráfico indican que la cantidad de recursos de red requerida en cada punto para separar flujos de tráfico puede ser limitada e independiente del tamaño de la red, siempre que dichos recursos se asignen y se liberen según sean o no necesarios en un momento dado.

- El mecanismo de encaminamiento de paquetes implementado en ciertas tecnologías hace posible, además, identificar un punto concreto de la red desde cualquier otro punto de la misma. Esto permite, por una parte, localizar con exactitud los puntos congestionados (no necesariamente terminales), y, por otra, saber con antelación si un paquete situado en un determinado punto de la red cruzará o no posteriormente por un punto congestionado.

En definitiva, se dan las condiciones que justifican plenamente un estudio como el presente: existe una necesidad crítica para la que no existe una respuesta satisfactoria, y existen ideas para desarrollar una solución adecuada, factible y realista.

1.4. Objetivos

A pesar de que la finalidad fundamental de la presente tesis, tal y como ya se ha explicado en los puntos anteriores, es la obtención de una técnica de control de congestión eficiente y escalable para redes de interconexión, es posible desglosar este objetivo global en una serie de objetivos parciales que se detallan a continuación:

1. Análisis general del fenómeno de la congestión en redes de interconexión, desde los mecanismos que originan su aparición y favorecen su evolución, hasta los efectos que puede tener su presencia en el funcionamiento de la red.

2. Estudio de las soluciones existentes actualmente para el control de congestión en redes de interconexión, determinando en qué circunstancias estas soluciones pueden presentar inconvenientes que les impidan responder adecuadamente a situaciones de congestión.

3. Diseño de una técnica de control de congestión para redes de interconexión que, basándose en el comportamiento del tráfico en situaciones de congestión y en las actuales tecnologías de red, solucione los problemas asociados a la congestión con la mayor eficiencia posible y mediante un número de recursos acotado.

4. Evaluación de la técnica diseñada mediante herramientas de simulación, en un número de configuraciones de topología y de tráfico suficiente para constatar su validez en cualquier situación.

5. Ajuste y optimización de la técnica diseñada, si se considerase necesario a partir de los resultados de su evaluación, hasta obtener una técnica definitiva que responda adecuadamente a las exigencias actuales de las redes de interconexión respecto al control de congestión.

El cumplimiento de los objetivos enumerados supondría alcanzar la meta planteada en nuestro estudio. El grado de consecución de dichos objetivos se analizará en el capítulo final de la presente memoria.

1.5. Estructura de la memoria de tesis

Una vez esbozados en el presente capítulo tanto el entorno en el que se sitúa nuestra propuesta como los beneficios que pretendemos conseguir con ella, es momento de explicar nuestro trabajo con mayor detalle. A ello están dedicados el resto de capítulos de esta memoria, estructurados del siguiente modo:

- **El capítulo 2** se ocupa de mostrar, de forma genérica, las características más relevantes de las redes de interconexión actuales, incidiendo tanto en los aspectos meramente tecnológicos como en aquéllos relativos a su utilización en distintos sistemas. Además, también se aborda una tecnología de red en particular, en concreto una de las más recientes propuestas de redes de interconexión: *PCI-Express Advanced Switching*.

- **El capítulo 3** ofrece una amplia visión de los distintos enfoques con los que se ha intentado solucionar anteriormente el problema de la congestión en redes de interconexión, concretando en diversas técnicas de las que se analizan en profundidad sus ventajas e inconvenientes.

- **El capítulo 4** explica con detalle nuestra propuesta inicial de técnica de control de congestión. Se expone, además, la metodología seguida en las pruebas para la evaluación de la técnica, especialmente las características de las herramientas empleadas. También se muestran, para distintos patrones de tráfico y topologías de red, los resultados que permiten la evaluación de esta técnica inicial, lo que permite extraer, para finalizar el capítulo, las primeras conclusiones sobre su validez.

- **El capítulo 5** está dedicado a los ajustes que hemos considerado necesario introducir en la propuesta inicial de cara a optimizar sus prestaciones y a dotarla de

más independencia ante las distintas formas de crecimiento y desaparición de los árboles de congestión. Como en el capítulo precedente, también en éste se muestran resultados con los que evaluar las prestaciones de esta versión mejorada de la técnica, y las conclusiones pertinentes.

- **El capítulo 6**, para finalizar, recoge las conclusiones globales del estudio, junto con las ideas con las que pretendemos continuar trabajando en esta línea en el futuro.

Capítulo 2

Redes de Interconexión

El presente capítulo repasa conceptos básicos relativos al entorno donde se enmarcan los trabajos descritos en esta memoria de tesis: las redes de interconexión. En primer lugar se muestra una panorámica de los distintos sistemas donde estas redes se emplean hoy en día, para a continuación pasar a ocuparnos de los aspectos más importantes en su diseño y funcionamiento: componentes, arquitecturas, técnicas de conmutación, de control de flujo, etc.. Obviamente, y teniendo en cuenta los objetivos de nuestra propuesta, dedicaremos especial atención a la congestión en redes de interconexión.

Conviene aclarar desde este punto que en todo momento nos centraremos en las modernas redes de interconexión de altas prestaciones, que, en general, se ajustan a un modelo básico común caracterizado por el uso de técnicas de conmutación, por el empleo de enlaces punto a punto de alta velocidad para conectar elementos, y por cierta flexibilidad y variabilidad en cuanto a las topologías posibles. Ahora bien, recordemos que las actuales redes de altas prestaciones son el resultado de una evolución de la tecnología, y han heredado algunas características tanto de las redes locales convencionales (en general de medio compartido) como de las "clásicas" redes de interconexión de grandes computadores paralelos (generalmente de topología regular y fija). Por tanto, algunos conceptos y técnicas reflejados en este capítulo también son propios de estas redes "tradicionales".

El capítulo finaliza analizando diversos aspectos de una tecnología de red de interconexión concreta: la especificación *PCI-Express Advanced Switching* [ASIb, ASIa]. De reciente aparición, esta tecnología resulta de especial relevancia para nuestro estudio al haber sido considerada desde el principio como ejemplo de plataforma donde podría implementarse realmente la técnica de control de congestión que se propone. Esto es así por cumplir plenamente esta especificación con los mínimos requisitos que plantea nuestra propuesta. Por ello, haremos especial hincapié en aquellas propiedades de *PCI-Express Advanced Switching* que volveremos a tratar más adelante en los capítulos dedicados al desarrollo de la nueva técnica de control de congestión.

2.1. Ámbito de aplicación

Hoy en día, las redes de interconexión de altas prestaciones se utilizan en distintos sistemas dedicados tanto a la computación como a la comunicación. Aun cuando la tecnología de estas redes permite, en general, su empleo en sistemas poco exigentes, como redes "domésticas" de área local o de sistema, sus altas prestaciones las convierten en una solución más adecuada para sistemas con grandes requisitos, bien en cuanto a capacidad de cómputo, bien en cuanto a velocidad de transmisión de datos. Estos sistemas podrían clasificarse en tres grandes grupos: *Clusters* de PCs y estaciones de trabajo, computadores masivamente paralelos y *routers* IP. En estos tres entornos se hace indispensable la utilización de tecnologías de interconexión eficientes, donde la red de interconexión tendrá un papel fundamental en las prestaciones finales del sistema, debiendo interconectar de forma eficiente hasta miles de nodos. A cada uno de estos tipos de sistemas están dedicados los puntos siguientes.

2.1.1. *Clusters* de PCs y estaciones de trabajo

Un *cluster* es, básicamente, un conjunto de ordenadores personales o estaciones de trabajo interconectados mediante una red. En la práctica, también suelen incluir otros componentes, como dispositivos de almacenamiento. En los *clusters* de altas prestaciones, es de gran importancia contar con un mecanismo de interconexión de alta velocidad, tanto entre los procesadores con los dispositivos, como entre los dispositivos entre sí. Obviamente, las redes de interconexión de altas prestaciones responden a esta necesidad.

Los *clusters* comenzaron a popularizarse a mediados de la década de los noventa del pasado siglo como una alternativa a los grandes sistemas multiprocesadores, debido principalmente a su mejor relación coste/prestaciones y a la mayor disponibilidad y facilidad de gestión y mantenimiento de sus componentes. Inicialmente, pues, los *clusters* se usaron básicamente como plataforma para la ejecución de las aplicaciones paralelas.

Pero, posteriormente, otro tipo de aplicaciones y servicios han encontrado en los *clusters* la mejor respuesta a sus necesidades. Por ejemplo, las redes de almacenamiento (*Storage Area Networks*, *STANs*), que requieren redes de interconexión de alta velocidad para garantizar tiempos de respuesta adecuados a aplicaciones y usuarios finales del sistema.

También es habitual actualmente que los *clusters* se empleen como base de servidores de Internet. Es ampliamente conocido que la gran difusión de Internet en los últimos años ha motivado la aparición y expansión de nuevos servicios basados en red, tales como la *WWW* y el comercio electrónico. Inicialmente, estos servicios los proporcionaban servidores basados en computadores personales de gama alta. Al aumentar el

volumen de accesos y la sofisticación de los servicios ofrecidos (por ejemplo, muchos de los servidores *WWW* ofrecen dinámicamente contenidos adaptados al perfil del usuario que se ha conectado) se ha hecho necesario reducir el tiempo de respuesta, aumentar la potencia de cálculo y aumentar la capacidad de almacenamiento de los servidores. Para proporcionar la capacidad de procesamiento y almacenamiento necesarias, muchos de estos grandes servidores se basan en la utilización de *clusters* de PCs.

Como ejemplos de importantes *clusters* en funcionamiento a día de hoy, cabe citar el *CPLANT* (*Computational PLANT*) [Rea99] implementado en los laboratorios *Sandia* [Lab]. Además, compañías como *Amazon, AOL, Google, Hotmail, Inktomi, WebTV* y *Yahoo* utilizan servidores basados en *clusters* de PCs o de estaciones de trabajo para proporcionar los servicios usados por millones de personas cada día [HP03]. En particular, y aunque se mantiene cierto secreto sobre la estructura exacta, se supone que el *cluster* de *Google* [BDH03] está compuesto por algo más de cien mil procesadores y dispone de una capacidad de almacenamiento de algunos miles de Terabytes. Otro ejemplo es el MCR Linux Cluster [lc]. Dicho sistema posee 1.152 nodos, cada uno con dos procesadores. Mucho más reciente es la puesta en marcha del supercomputador *MareNostrum* [Mar], que puede alcanzar una capacidad de cómputo de 40 Teraflops. Este supercomputador cuenta con 2.406 nodos de procesamiento con dos procesadores cada uno, y con una capacidad de almacenamiento total de 240 Terabytes. La red empleada para la interconexión está compuesta por conmutadores de tecnología *Myrinet* [BCF95, Inc]. A día de hoy, este supercomputador está ubicado en la octava posición en la lista de los 500 supercomputadores más potentes del mundo (*TOP500 list*) [lis].

2.1.2. Computadores masivamente paralelos

Aunque la evolución de la tecnología ha aumentado la versatilidad de las redes de interconexión de altas prestaciones, permitiendo su uso en nuevos tipos de sistemas, no por ello han dejado de emplearse en las "clásicas" estructuras de los computadores masivamente paralelos (multicomputadores y multiprocesadores). En estos sistemas, la red debe ofrecer unas latencias de comunicación reducidas para evitar tener bloqueados los procesos un tiempo excesivo. Además, debe proporcionar un elevado ancho de banda con el fin de permitir la comunicación concurrente entre todos los procesos.

La demanda de estos sistemas es debida a que existen ciertos niveles de potencia de cálculo que sólo ellos pueden alcanzar. Por ejemplo, existen muchas aplicaciones de cálculo intensivo que requieren una elevada potencia de procesamiento: plegado de proteínas, simulaciones de reacciones nucleares, modelado climático, simulaciones de interacciones de galaxias, etc.. Es más, simplemente incrementando el nivel de detalle del análisis o simulación (por ejemplo, simulando el plegado de proteínas durante 10 milisegundos en vez de 1 milisegundo e interaccionando con un mayor número de moléculas de agua, varios órdenes de magnitud) estas aplicaciones requerirán de una

mayor potencia de cálculo. A consecuencia de ello, estas aplicaciones demandan un desarrollo continuado en la tecnología (y en la investigación asociada) para la obtención de computadores con mayores potencias de cálculo.

Ejemplos de estos sistemas son el *Earth Simulator* [SC] con 640 nodos de procesamiento vectoriales (cada uno de ellos con 8 *pipelines*), el *ASCI Red* [Sup] con 4.512 nodos de computación, y el *BlueGene/L* [Tea02] con 65.536 nodos de computación (cada uno de ellos con dos procesadores).

2.1.3. *Routers* IP

Desde hace años, existe un creciente interés en la construcción de encaminadores de altas prestaciones para tráfico IP (*routers* IP) [Aea92, Pea98, Bak95]. Esto es debido a que el espectacular crecimiento del número de usuarios de la red ha provocado una enorme demanda de ancho de banda. De hecho, se calcula que, actualmente, esta demanda se dobla cada año [Pea99]. Las líneas de comunicación (enlaces) que transmiten el tráfico IP por todo el mundo han podido responder a estas necesidades crecientes de ancho de banda gracias al uso de las tecnologías de fibra óptica y al uso de distintas técnicas de multiplexación, como la basada en la división de onda (*wavelength-division multiplexing, WDM*) [Muk97, RS98].

Pero en el caso de los elementos de conmutación de la red, los *routers* IP, no ha sido fácil conseguir su adaptación a la nueva situación. Debido a las limitaciones impuestas por la tecnología con la que se fabrican estos conmutadores (VLSI), sólo se ha podido conseguir (aproximadamente) doblar sus prestaciones cada dieciocho meses. Por tanto, existe un desfase entre el ancho de banda demandado por los usuarios y el ofrecido por los *routers*, lo que provocaría "cuellos de botella" en los elementos de conmutación, que no podrían atender al total del tráfico entrante por los enlaces.

Para solucionar este problema, se ha tendido a aumentar el número de puertos de los *routers*, para que de este modo no se vean desbordados por el creciente tráfico IP. Así, el número de puertos de los *routers* ha crecido, y pronto será habitual encontrar *routers* con miles de puertos. A menos que se descubran técnicas eficientes para realizar la conmutación usando exclusivamente tecnología óptica, la única opción para la construcción de *routers* con tal número de puertos es que éstos adopten internamente una estructura formada por una o varias etapas de elementos de conmutación (*Switch Fabric*). En estos casos, las redes de interconexión de altas prestaciones son una elección evidente a la hora de confeccionar *Switch Fabrics*. Un ejemplo de *router* con esta estructura es el *AVICI Terabit Router* [DCD98].

2.2. Componentes y topologías

En general, una red de interconexión de altas prestaciones actual está compuesta por varios conmutadores interconectados entre sí mediante enlaces punto a punto. Así pues, podemos considerar como componentes esenciales de una red a sus conmutadores y enlaces. Habitualmente, los enlaces también se emplean para que los terminales (nodos de procesamiento o dispositivos de almacenamiento) se conecten a la red, a través de interfaces de red o adaptadores de entrada y salida (*input/output adapters*). Aunque desde cierto punto de vista dichos interfaces podrían considerarse también componentes de la red, en los siguientes puntos nos centraremos exclusivamente en el funcionamiento y características básicas de los componentes esenciales.

2.2.1. Enlaces

Como ya se ha indicado anteriormente, en las redes de interconexión modernas la comunicación entre dos dispositivos se realiza habitualmente mediante un enlace **punto a punto**. Al ser frecuente que los dispositivos que se comunican tengan relojes independientes, la transmisión de datos a través del enlace se suele realizar de forma **asíncrona**. En consecuencia, se necesita emplear algún protocolo que garantice que la comunicación se ha realizado correctamente.

Una de las cualidades más importantes de un enlace es su **ancho de banda**. El ancho de banda del enlace es el producto de la frecuencia de reloj del enlace y de la "anchura" del canal (número de bits que se transmiten en paralelo). Evidentemente, el ancho de banda de un enlace debe ser lo mayor posible para conseguir las mejores prestaciones. La tecnología empleada hoy en día en los enlaces permite que éstos tengan elevadas frecuencias de reloj, aumentando así el ancho de banda. Actualmente, los enlaces de las redes de interconexión alcanzan anchos de banda del orden de Gigabits por segundo (*Gbps*). Por ejemplo, el ancho de banda de los enlaces de la última versión de *Myrinet* es de 10 *Gbps*, mientras que el ancho de banda base de los enlaces de *Infiniband* [IBA], es de 2,5 *Gbps*.

Ahora bien, dado que los datos no se propagan instantáneamente a través de los cables, para aprovechar estos grandes anchos de banda suele ser necesario inyectar datos en el enlace antes de que los enviados previamente hayan alcanzado el otro extremo. En este caso, múltiples bytes pueden encontrarse "en vuelo" en el enlace al mismo tiempo. Esta técnica es conocida como **segmentación de canales** (*channel pipelining*) [SG94]. Además, para que un enlace sea realmente eficiente, se requiere que sea de tipo ***full-duplex*** (la comunicación puede realizarse en ambos sentidos del enlace simultáneamente) en lugar de ***half-duplex*** (la comunicación puede realizarse en ambos sentidos pero no simultáneamente).

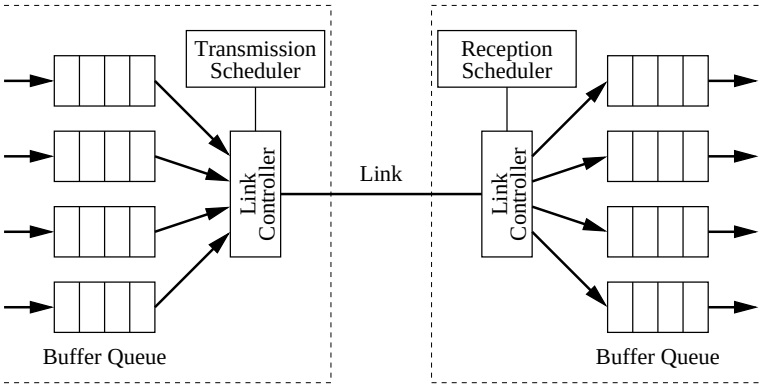


Figura 2.1: Estructuras implicadas en la transmisión de datos a través de un enlace punto a punto.

Existen ciertas estructuras que, sin formar parte de los enlaces, están directamente asociadas al proceso de transmisión de datos a través de los mismos. Por ejemplo, generalmente, el emisor almacena los datos a transmitir en algún tipo de cola (***buffer***), y lo mismo sucede en el receptor tras la llegada de dichos datos. Estos *buffers* pueden consistir en una simple cola con estructura *FIFO* (*first in, first out*), aunque muchas otras implementaciones son también posibles [TF88]. Esta diversidad de políticas ha sido debida sobre todo a la flexibilidad que proporciona la implementación de *buffers* en memoria RAM [EM00]. En cualquier caso, es necesario garantizar que la capacidad del *buffer* del receptor no se desborde. Esto se consigue mediante el adecuado **control de flujo**, del que nos ocuparemos con más detalle en un punto posterior.

La mencionada libertad ofrecida por la RAM para la gestión de *buffers* es especialmente útil para multiplexar varios flujos de información sobre el mismo enlace físico. En estos casos, paquetes pertenecientes a distintos flujos de información se almacenarán en *buffers* distintos tanto en el emisor como en el receptor. Sin embargo, esto hace necesario disponer de un **planificador** (*transmission scheduler*) adecuado en el emisor que seleccione qué *buffer* tendrá acceso al enlace cada vez que éste se libere. Además, hay que garantizar que la información enviada se almacenará en el *buffer* correcto en el lado del receptor. En aquellos entornos que ofrecen garantías de calidad de servicio (QoS), algunos flujos de información requieren que se les garantice un mínimo ancho de banda y/o un máximo retardo en la transmisión, lo que acarrea una mayor complejidad en el planificador.

En la figura 2.1 se muestran las estructuras que intervienen en la transmisión de datos a través de un enlace punto a punto, en el caso de contar con varios *buffers* tanto en el emisor como en el receptor.

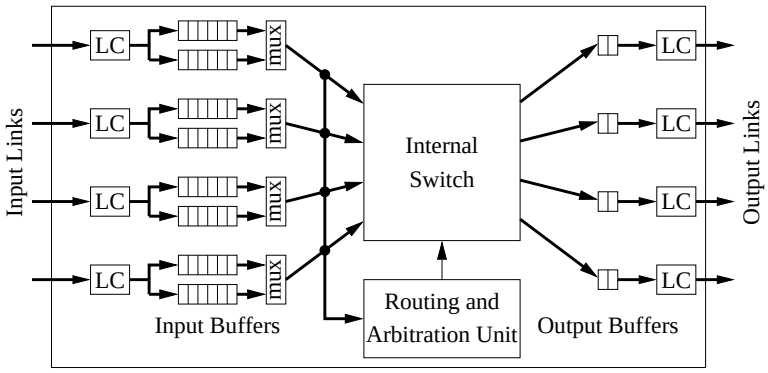


Figura 2.2: Esquema de bloques de un conmutador típico.

2.2.2. Conmutadores

Un conmutador (*switch*) es un dispositivo capaz de interconectar entre sí varios dispositivos. La figura 2.2 muestra el esquema de bloques de un conmutador típico, con las estructuras que habitualmente lo componen.

Como se aprecia en el esquema, los puertos de entrada y salida se conectan a unos enlaces externos, para permitir la recepción y el envío de datos desde y hacia el exterior del conmutador. A su vez, los datos en tránsito se almacenan en los *buffers* de los puertos de entrada y salida. Cuando existen múltiples *buffers* en los puertos, es necesario determinar cuál es el *buffer* donde deben almacenarse los datos. Ahora bien, la presencia simultánea de *buffers* en los puertos de entrada y salida no es imprescindible para el funcionamiento del conmutador. Dependiendo de la existencia o no de *buffers* en los puertos de entrada y salida, los conmutadores se clasifican en:

- ***OQ (Output Queuing) switches***, con *buffers* sólo en los puertos de salida.
- ***IQ (Input Queuing) switches***, con *buffers* sólo en los puertos de entrada.
- ***CIOQ (Combined Input and Output Queuing) switches***, con *buffers* tanto en los puertos de entrada como en los de salida.

Cabe destacar que existen otras posibilidades respecto a la situación de los *buffers* en el conmutador. Concretamente, los *BC (Buffer Crossbar) switches* [RCOC01] disponen de *buffers* en los puntos de cruce del *crossbar*. Además, también existen propuestas "híbridas", como los *HC (Hierarchical Crossbar) switches* [KDT05], que surgen de una combinación de los tipos *CIOQ* y *BC*.

El elemento central del conmutador es el **conmutador interno** (*internal switch*). Su función es conectar los puertos de entrada con los de salida, permitiendo la transferencia de datos entre unos y otros. En la mayoría de conmutadores, este elemento se

implementa mediante una red ***crossbar***. Las entradas y salidas del *crossbar* pueden estar **demultiplexadas**, pero esto complica bastante la implementación física del conmutador, por lo que en la mayoría de redes comerciales, el *crossbar* está **multiplexado** tanto a la entrada como a la salida.

Una propiedad fundamental del conmutador interno es el valor de su ***speedup***, que indica la "aceleración" en la transferencia de datos a través del conmutador respecto a la velocidad a la que estos datos llegan al mismo. Así, un conmutador con un *speedup* de S es capaz de transferir S unidades de información desde uno o varios puertos de entrada a un puerto de salida, en el periodo de tiempo que tarda en llegar cada unidad de información a los puertos de entrada. Nótese que este parámetro está relacionado con la estructura de *buffers* del conmutador, ya que, por ejemplo, un conmutador OQ de dimensiones $N \times N$ (o sea, con N puertos de entrada y N de salida) necesita un *speedup* interno de N para poder aceptar todo el tráfico entrante en el peor caso. Por otra parte, un conmutador IQ necesita un *speedup* de 1.

Desde hace algún tiempo, el aumento de la velocidad de los enlaces y el incremento del número de puertos de los conmutadores dificultan enormemente la implementación de esquemas OQ, que en estas condiciones requerirían velocidades de conmutación prácticamente inalcanzables. Debido a esto, la tendencia actual es usar conmutadores $CIOQ$, que, teóricamente, permiten valores de *speedup* entre 1 y N. En general, el aumento del *speedup* incrementa notablemente el coste del conmutador, por lo que los conmutadores comerciales suelen presentar valores de *speedup* entre 1,5 y 2.

Por su parte, la **unidad de control** del conmutador (o unidad de encaminamiento y arbitraje, *routing and arbitration unit*) se encarga de realizar otras tareas fundamentales para el funcionamiento del conmutador. Una de ellas es el **encaminamiento** (*routing*), mediante el cual se determina hacia qué puerto de salida deben transferirse los datos que llegan a un puerto de entrada en un momento dado. La unidad de control no puede realizar esta función sin contar con cierta información, para lo cual los datos se agrupan en **paquetes**, que además contienen en su cabecera algún tipo de información de encaminamiento que puede ser procesada por la unidad de control para calcular el puerto de salida solicitado en cada caso. La forma en que esta información es generada y procesada da lugar a distintas **técnicas y algoritmos de encaminamiento**, a los que se dedica un punto posterior.

Dependiendo de la técnica de encaminamiento implementada en la red, el procesado de la información de encaminamiento contenida en la cabecera de un paquete puede dar como resultado un conjunto de puertos de salida hacia los que es posible encaminarlo. En estos casos, la unidad de control también debe realizar una función de **selección** de un puerto concreto de entre todos los posibles.

En ocasiones, varios paquetes situados en los puertos de entrada solicitan a la vez cruzar hacia el mismo puerto de salida. Habitualmente, no todas estas peticiones de

cruce podrán ser atendidas simultáneamente, por lo que alguno o algunos de los paquetes solicitantes deberán esperar. Este fenómeno es conocido como **contención**, y en casos extremos puede dar lugar a la **congestión**, que será extensamente analizada posteriormente. En situaciones de contención se hace imprescindible otra importante función realizada por la unidad de control: el **arbitraje** (*arbitration*), mediante el cual se selecciona una de entre todas las peticiones existentes para cruzar a un determinado puerto de salida. Aquellos paquetes cuyas peticiones no se atienden se consideran **bloqueados** y, en la mayoría de redes de interconexión actuales, no se descartan (redes "sin pérdidas"), sino que esperan hasta que su petición es atendida. El arbitraje puede realizarse con diferentes criterios [GMA02], como por ejemplo atendiendo las peticiones de los puertos por turnos cíclicos (*round-robin*) o dando preferencia a las peticiones de paquetes con mayor tiempo de espera.

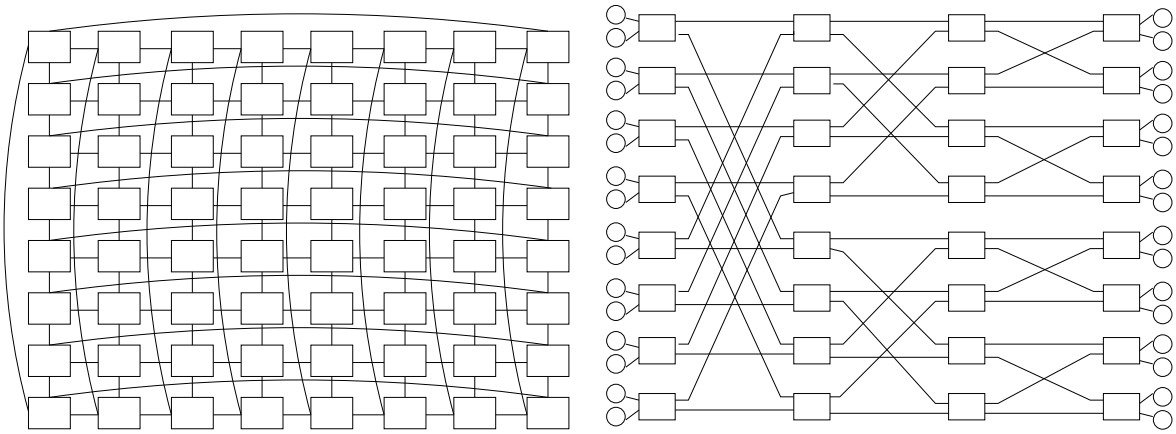
Una vez conocido el puerto de salida solicitado por un paquete, y aceptada su petición de cruce, existen diferentes enfoques para establecer cuándo y cómo se produce el avance de los datos incluidos en el paquete a través del conmutador interno. Esto viene determinado por la **técnica de conmutación** empleada, como veremos más adelante.

Por último, cabe destacar que, igual que ocurre con la transmisión de datos a través de un enlace, también diferentes flujos de información pueden tener diferentes prioridades de cara a atravesar el conmutador (de nuevo, esto suele suceder en entornos donde existen flujos de datos con requisitos de calidad de servicio), lo cual debe ser tenido en cuenta por la unidad de control al realizar sus distintas funciones [ASD04].

2.2.3. Topologías

Habitualmente, un único conmutador no es suficiente para conectar todos los terminales y/o dispositivos del sistema, por lo que la red contará, en general, con varios conmutadores conectados entre sí. La **topología** de la red define el modo en que se interconectan los conmutadores que forman parte de la misma. Una condición exigible a la topología es que sea totalmente **conexa**, es decir, que permita la conexión entre todos los terminales y dispositivos del sistema.

Se dice que una topología es **regular** cuando las conexiones entre conmutadores siguen un patrón de conexión regular y definido (ver figura 2.3). En estos casos, una red de interconexión puede generarse mediante la aplicación de este patrón, de modo que basta con dimensionar la red para que quede totalmente definida su topología exacta. Existen numerosos patrones cuyas propiedades han sido ampliamente estudiadas (toros, redes multietapa, hipercubos, etc.), y se han usado desde hace tiempo en redes regulares reales. Las topologías regulares se emplean generalmente en los computadores masivamente paralelos.



(a) (b)

Figura 2.3: Redes de topología regular: Torus (a) y multietapa butterfly (b).

En el caso de los *clusters*, dados su distribución espacial y su carácter flexible, la conexión entre conmutadores puede no seguir un patrón definido (salvo en aquellos *clusters* dedicados a computación de altas prestaciones), empleándose entonces una topología **irregular**. El uso de topologías irregulares complica bastante, en general, diversos aspectos del funcionamiento de la red (como el encaminamiento), pero proporciona una mayor libertad de uso, gestión y expansión. Además, conviene tener presente que una topología regular se convierte en irregular en caso de fallo de alguno de sus conmutadores o enlaces.

Además de definir el patrón seguido en la interconexión de los conmutadores, la topología determina también el número de terminales o dispositivos conectados directamente a ellos. En el caso de los computadores masivamente paralelos, los conmutadores suelen tener conectado un único terminal, mientras que en los *clusters* pueden tener conectados un número variable de terminales.

2.3. Técnicas de control de flujo

Como ya se ha mencionado previamente, es práctica común que a ambos extremos de un enlace tanto el transmisor como el receptor incorporen algún tipo de *buffer* para contener los datos enviados y recibidos, por lo que es necesario implementar algún mecanismo de control de flujo para garantizar que la capacidad del receptor no se desborde.

Mediante el control de flujo, pues, se notifica al emisor la disponibilidad o no de espacio en el receptor. La unidad de control de flujo es la cantidad de información "atómica" considerada por el mecanismo de control de flujo. Dicho de otro modo, es

la cantidad máxima de información que puede transmitirse ininterrumpidamente sin que el mecanismo de control de flujo pueda evitarlo. La capacidad de los *buffers* suele medirse en unidades de control de flujo.

En los computadores paralelos, el control de flujo se implementa con señales de control adicionales [DS86]. Esto es posible ya que tradicionalmente los enlaces en estos sistemas son cortos y prácticamente todos de la misma longitud. Sin embargo, en los *clusters* no sucede así, ya que los enlaces suelen ser largos (hasta 18 metros en *Myrinet*) y de distinto tamaño, por lo que en estos casos se emplean mecanismos de control de flujo basados en el envío de mensajes de control.

Uno de estos mecanismos es el **control de flujo basado en créditos** [Com], donde cada conmutador recibe inicialmente varios créditos para enviar mensajes a un conmutador vecino. En cada transmisión de un paquete, el emisor consume un crédito, deteniendo la emisión de paquetes cuando ha consumido todos sus créditos. Cuando se libera espacio en el receptor, se envían nuevos créditos al emisor. Este tipo de control de flujo es el empleado, por ejemplo, por *Infiniband* y *Advanced Switching*.

Otra solución es el **control de flujo por medio de marcas**, como en el control de flujo *Stop & Go* [BCF95] empleado por *Myrinet*. En este caso, cuando el receptor detecta que su *buffer* de entrada está próximo a desbordarse (su nivel excede de la marca *Stop*), envía al emisor un símbolo de *STOP* para que éste detenga el envío de paquetes. Cuando el receptor tiene espacio suficiente en su *buffer* (su nivel desciende por debajo de la marca *Go*), envía al emisor un símbolo de *GO* para que se restablezca el envío de paquetes. *Advanced Switching* también contempla (opcionalmente) el uso de este tipo de control de flujo, denominado en este caso *Xon/Xoff*.

2.4. Técnicas de conmutación

La *técnica de conmutación* determina cómo se produce el avance de las unidades de control de flujo contenidas en un paquete entre los puertos de entrada y salida de un conmutador. Las tres técnicas de conmutación más utilizadas en las redes actuales son *store & forward*, *virtual cut-through* y *wormhole*.

- Cuando se usa ***store & forward*** [Tan96], el paquete se almacena totalmente en el *buffer* asociado al puerto de entrada antes de ser reenviado hacia el puerto de salida. Con este mecanismo, la latencia del paquete es proporcional al número de conmutadores atravesados, al tiempo que se limita el tamaño máximo de los paquetes.

- En la conmutación ***virtual cut-through*** [KK79] un paquete se retransmite hacia el siguiente conmutador tan pronto como la cabecera es decodificada, sin

necesidad de que se reciba previamente todo el paquete. No obstante, si el canal de salida seleccionado está ocupado, el paquete se almacena por completo en el *buffer* de entrada hasta que la salida esté disponible. Por lo tanto, esta técnica se comporta como *store & forward* cuando el canal de salida está ocupado. La latencia del paquete se ve poco influenciada por la distancia al destino. Sin embargo, el tamaño de los paquetes está limitado. Tanto *Infiniband* como *Advanced Switching* emplean este tipo de conmutación.

- En el caso de ***wormhole*** [DS87], igual que en *virtual cut-through*, el paquete se reenvía inmediatamente hacia el puerto de salida antes de recibirlo por completo. Sin embargo, en caso de que el puerto de salida esté ocupado, el paquete puede no almacenarse completamente en el *buffer* asociado al puerto de entrada, sino que puede quedar almacenado a lo largo de todos los *buffers* de los conmutadores visitados. Conforme los *buffers* visitados queden sin espacio, el protocolo de control de flujo bloqueará el avance del paquete. Con este mecanismo, el tamaño del paquete no está limitado, requiriendo además poco espacio de almacenamiento en los *buffers* asociados a los puertos de entrada de los conmutadores. Este tipo de conmutación es el empleado por *Myrinet*.

2.5. Encaminamiento

Como hemos visto, las redes de interconexión actuales están generalmente compuestas por múltiples conmutadores, por lo que un paquete, para llegar a su destino, necesitará seguir una ruta en la que cruzará por un número variable de enlaces punto a punto y conmutadores. Es necesario, pues, que existan mecanismos para que en todo momento el paquete avance siguiendo una ruta válida que le permita alcanzar correctamente su destino. Para ello, la cabecera del paquete debe contener información para que éste sea encaminado en todos los conmutadores de forma correcta. Esta información debe generarse y procesarse siguiendo cierta planificación o estrategia definida en un **algoritmo de encaminamiento**. La aplicación a la topología concreta de la red de un algoritmo de encaminamiento nos daría la **función de encaminamiento** de la red. Esta función devuelve la ruta (o rutas) para cada par origen-destino de un paquete.

Como en el caso de la topología, es naturalmente deseable que la función de encaminamiento sea **totalmente conexa**, es decir, que ofrezca al menos una ruta válida para cada posible pareja origen-destino. Otra cualidad esencial de la función de encaminamiento es que ofrezca garantía de ausencia de interbloqueos (*deadlock freedom*). Una situación de interbloqueo (***deadlock***) (ver figura 2.4) se origina cuando varios paquetes se encuentran, de forma permanente, en una posición en la que no pueden avanzar hacia su destino porque solicitan recursos (*buffers*, canales, etc.) ya ocupados por otros

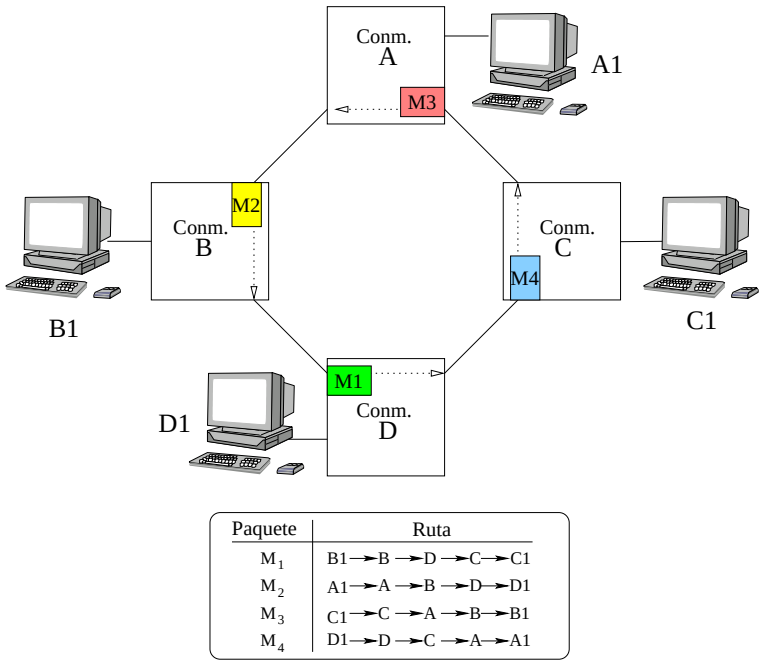


Figura 2.4: Situación de *deadlock* en una red de 4 conmutadores.

paquetes. También deben evitarse en el encaminamiento situaciones de ***livelock***, que se producen cuando algunos paquetes se encaminan continuamente sin alcanzar jamás su destino. Todos los problemas mencionados deberían evitarse diseñando (o seleccionando) cuidadosamente el algoritmo de encaminamiento [DYN02] y su aplicación a la red. En general, la resolución de estos problemas pasa por la imposición de ciertas restricciones en el encaminamiento [Dua93], a costa de rebajar en cierta medida las prestaciones de la red.

Los algoritmos de encaminamiento para redes de interconexión regulares resultan relativamente sencillos tanto en su diseño como en su implementación, pero en redes irregulares surgen diversas complicaciones. En cuanto al diseño, hay que considerar que un algoritmo para redes irregulares debe ser aplicable a cualquier topología y, por tanto, no se puede contar con patrones de conexión definidos, como en redes regulares. En cuanto a la implementación, hay que considerar que en redes regulares la función de encaminamiento será en general sencilla, y por tanto es posible su implementación *hardware*, pero en redes irregulares, esto sería una tarea compleja (máxime teniendo en cuenta que esta implementación puede ser distinta para cada nodo de la red). Una alternativa a la implementación *hardware* de la función de encaminamiento es reflejar dicha función de forma explícita en **tablas de encaminamiento** (*routing tables*). Mediante una consulta en la tabla se decidirá, dado el destino y origen de un paquete (y, en ocasiones, dado el puerto de entrada al conmutador), la ruta a seguir por el mismo.

Algunos ejemplos de los algoritmos de encaminamiento más conocidos y sencillos de implementar, que además ofrecen garantías de ausencia de bloqueos, son el algoritmo *Up*/Down** [Sea91] y los algoritmos de tipo *Dimension Order Routing* [DYN02] (por ejemplo, el encaminamiento *X-Y* [DS87]).

La forma concreta de implementar el encaminamiento en una red de interconexión puede definirse como **técnica o mecanismo de encaminamiento** de la misma. Dependiendo de qué elementos de la red tienen acceso a la función de encaminamiento, se establece la siguiente clasificación de las técnicas de encaminamiento:

- **Encaminamiento fuente.** En este caso, sólo los terminales tienen acceso a la función de encaminamiento. A partir de ésta, toda la ruta a seguir por el paquete se calcula en el terminal que lo genera. Tras calcular la ruta, el terminal la "escribe" de algún modo en la cabecera del paquete. En general, la ruta se compondrá de varios fragmentos de información de encaminamiento, estando cada fragmento destinado a un conmutador de la ruta. Para encaminar un paquete, cada conmutador "lee" la cabecera del mismo y utiliza la información correspondiente para determinar el puerto (o puertos) de salida solicitado. Es importante resaltar que con este mecanismo los conmutadores no pueden tomar decisiones de encaminamiento (o, lo que es lo mismo, no tienen acceso a la función de encaminamiento), sino que sólo interpretan la información contenida en la cabecera y actúan en consecuencia. En algunas tecnologías (como *Myrinet*), una vez encaminado el paquete por un conmutador, se elimina de la cabecera la información destinada a dicho conmutador, pero en otras (como *PCI-Express Advanced Switching*), se conserva la ruta completa en la cabecera durante todo el trayecto del paquete.

- **Encaminamiento distribuido.** Si se usa encaminamiento distribuido, la única información de encaminamiento que contiene la cabecera del paquete es el identificador del destino. A partir de esta información, cada conmutador debe calcular el puerto (o puertos) de salida solicitado por el paquete. Esto significa que, en este caso, los conmutadores deben tomar decisiones de encaminamiento, y para ello deben tener acceso a la función de encaminamiento, bien de forma implícita, mediante una implementación *hardware* de la misma, bien de forma explícita, mediante consultas a una tabla de encaminamiento. Ejemplos de tecnologías de red que usan encaminamiento distribuido son *Autonet* [Sea91] e *Infiniband*.

Para muchas topologías, pueden existir varios caminos posibles entre los dos dispositivos que pretenden comunicarse. En estos casos, la función de encaminamiento puede considerar como válidas no una, sino varias rutas para una serie de parejas origen-destino. Respecto a la cantidad de rutas válidas que puede devolver la función de encaminamiento, las técnicas de encaminamiento se dividen en:

- **Encaminamiento determinista**. La función de encaminamiento devuelve sólo una ruta válida para cada par origen-destino. Esta estrategia suele tener un retardo de encaminamiento inferior, pero la contención en la red es mayor, y la saturación se alcanza con niveles de carga más bajos.

- **Encaminamiento adaptativo**. En este caso, la función de encaminamiento devuelve varias rutas válidas para encaminar un paquete, debiéndose seleccionar la más adecuada en cada caso, con arreglo a ciertos criterios [GY93, TLM03]. Por ejemplo, la posibilidad de utilizar varios caminos puede emplearse bien para evitar zonas congestionadas, bien para dar servicio en presencia de fallos. A su vez, las estrategias de **equilibrado de carga** [FGL99, SDT04] se centran en distribuir el tráfico entre todos los posibles caminos alternativos, de tal forma que la utilización de los enlaces sea uniforme. La forma más habitual de realizar dicho equilibrado es la selección de aquel camino que tiene una mayor cantidad de recursos disponibles. Ahora bien, el empleo de encaminamiento adaptativo puede generar ciertos problemas, como la entrega fuera de orden de los paquetes, lo cual no es aceptable en la mayoría de sistemas.

Por último, cabe reseñar que existen otros mecanismos directamente relacionados con el encaminamiento y que también suelen ser imprescindibles para el funcionamiento de la red. Un ejemplo serían los **mecanismos de reconfiguración** [ABC99] para redes irregulares, encargados de descubrir la topología de la red, calcular las tablas de encaminamiento y actualizarlas si fuese necesario tras detectar cambios en la topología.

2.6. Contención y congestión

La finalidad de los trabajos abordados en la presente tesis es desarrollar una técnica viable que solucione de forma eficiente uno de los problemas que pueden surgir en las redes de interconexión: la congestión. Ya hemos mencionado brevemente dicho problema en secciones anteriores, pero, al ser el centro de nuestros estudios, merece ser explicado detenidamente. Los siguientes puntos se ocupan de explicar el origen, la formación y las consecuencias de la congestión en redes de interconexión.

2.6.1. Contención en redes de interconexión

Como ya hemos mencionado, una de las funciones fundamentales de la unidad de control de un conmutador es el arbitraje, que se lleva a cabo cuando dos o más paquetes solicitan cruzar hacia un mismo puerto de salida desde sus respectivos puertos de entrada. Normalmente, en estos casos, la unidad de control debe seleccionar (siguiendo algún criterio de arbitraje) qué paquete de entre los solicitantes cruzará hacia el puerto

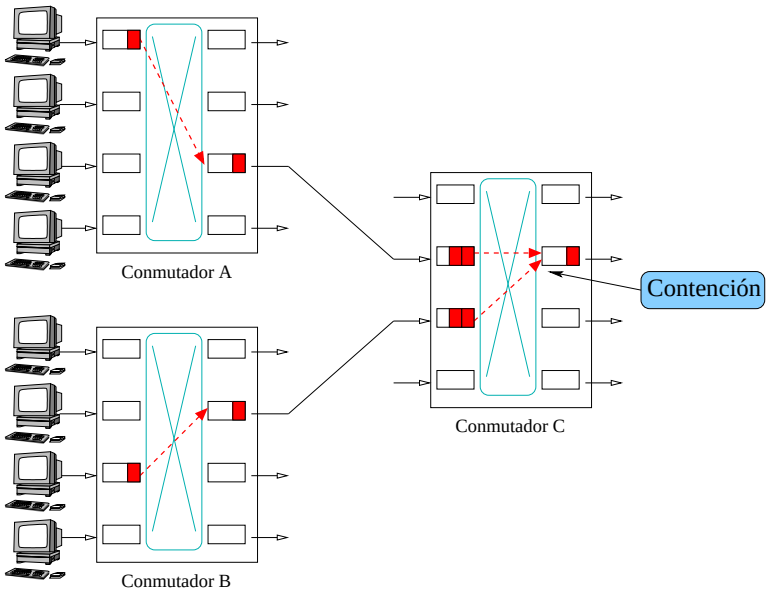


Figura 2.5: Contención entre paquetes solicitando un mismo puerto de salida.

de salida requerido, mientras que los paquetes no atendidos esperarán su turno. Se dice entonces que existe entre los paquetes implicados una situación de **contención** por un puerto de salida concreto.

La figura 2.5 muestra un ejemplo simple de contención. En este ejemplo (y en todos los de esta sección) se asume el uso de conmutadores *CIOQ*, y también que, tanto en los puertos de entrada como en los de salida de los mismos, sólo se cuenta con un *buffer* con política *FIFO*. Además, se supone que el conmutador interno ofrece un *speedup* de valor 1. Por último, se utiliza en estos esquemas la convención de que las flechas de trazo discontinuo apuntan al puerto de salida solicitado por los paquetes almacenados en el *buffer* del puerto de entrada donde se sitúa el origen de la flecha.

Como puede verse en la figura 2.5, existen dos flujos de paquetes con distintos orígenes (terminales conectados a los conmutadores A y B, respectivamente), que confluyen en el conmutador C, solicitando en éste un mismo puerto de salida desde distintos puertos de entrada. En este caso, en un instante dado, sólo un paquete podrá avanzar hacia el puerto de salida solicitado desde uno de los puertos de entrada, quedando bloqueado el paquete que solicite el cruce desde el otro puerto de entrada. Por tanto, se produce en el conmutador C una situación de contención por un puerto de salida. En los conmutadores A y B no existe contención al no existir dos paquetes que soliciten simultáneamente un mismo puerto de salida.

La contención es un fenómeno totalmente habitual en redes de interconexión. En la mayoría de las redes actuales, en una situación de contención, el paquete que no se selecciona para cruzar hacia el puerto de salida requerido no se descarta, sino que permanece almacenado en el *buffer* correspondiente a la espera de ser atendido.

Nótese que esto implica que el paquete estará bloqueado en el *buffer* hasta que pueda cruzar, y que los paquetes que lleguen mientras tanto a dicho *buffer* tampoco podrán avanzar debido al habitual carácter *FIFO* del mismo. Nótese también que, aunque el conmutador interno ofreciera cierto *speedup*, en las redes reales éste suele ser menor que 2. Es decir, antes de que dos paquetes contendientes puedan cruzar totalmente el conmutador interno, es perfectamente posible que lleguen a sus respectivos *buffers* nuevos paquetes. Evidentemente, esto será mucho más probable cuando el número de paquetes implicados en la contención desde distintos puertos de entrada sea mayor. Por tanto, mientras dure la contención y sigan llegando nuevos paquetes a los *buffers* que almacenan a los paquetes contendientes, es normal que se produzca una acumulación de paquetes en dichos *buffers*, aumentando su nivel de ocupación. Puede observarse que la figura 2.5 también refleja esta mayor ocupación en aquellos *buffers* de entrada del conmutador C implicados en la contención.

2.6.2. Congestión en redes de interconexión

El problema de la congestión surge cuando la contención persiste en el tiempo. Si esto sucede, la acumulación de paquetes en los *buffers* implicados puede crecer hasta ser excesiva, llegándose incluso a agotar completamente el espacio de memoria destinado a dichos *buffers*. En estas circunstancias, las prestaciones de la red acabarán por degradarse. Por ejemplo, la productividad de la red descenderá, y el retardo introducido por el tiempo de espera de los paquetes en los *buffers* congestionados aumentará su latencia hasta extremos intolerables para muchos tipos de tráfico. Evidentemente, esto afectará a las prestaciones del sistema en su conjunto, aumentando, por ejemplo, los tiempos de ejecución de las aplicaciones soportadas por el sistema y el tiempo de respuesta a los usuarios del mismo.

La figura 2.6 muestra la misma red que la figura 2.5, pero en un instante posterior en el que la intensidad de los flujos de datos que confluyen en el conmutador C ha prolongado en el tiempo la contención por el puerto de salida, originándose la congestión que puede apreciarse en los *buffers* de los puertos de entrada implicados en la contención.

Ahora bien, la situación se agrava todavía más cuando, debido a la elevada ocupación de los *buffers* congestionados, el control de flujo actúa e impide a *buffers* situados en otros conmutadores enviar nuevos paquetes (ver figura 2.6). Aunque estos paquetes ni siquiera habrán llegado al punto donde se produce la contención, también quedarán bloqueados por la acción del control de flujo en sus respectivos *buffers*. Éstos, por tanto, también pueden comenzar a sufrir una acumulación de paquetes si los flujos de datos son suficientemente intensos, congestionándose a su vez y obligando de nuevo al control de flujo a impedir el envío de paquetes desde otros *buffers*. Se produce de este modo y mediante el control de flujo una propagación de la congestión "hacia atrás" o

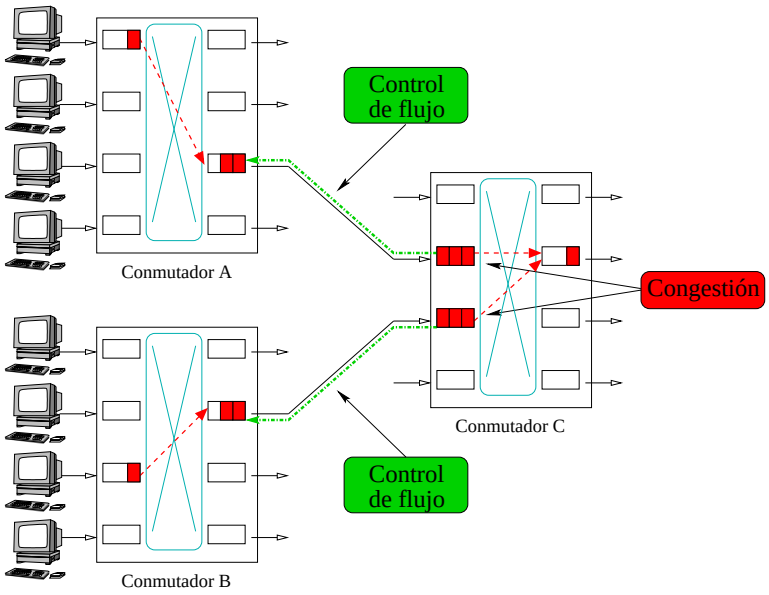


Figura 2.6: Congestión en la red por persistencia de la contención.

contracorriente ("*upstream*"), es decir, en el sentido contrario al avance de los flujos de datos (que se produce en sentido "*downstream*").

Así pues, en situaciones de congestión puede formarse a lo largo de varios conmutadores de la red una serie de *buffers* congestionados cuyo origen es una contención muy intensa en un punto de la misma. A estas series de *buffers* congestionados se les suele llamar **"árboles de congestión"**. Generalmente, el punto donde se origina un árbol de congestión (el punto donde sucede la contención original) recibe el nombre de **raíz** del árbol. Dado que la propagación de la congestión desde un puerto de salida puede afectar a varios puertos de entrada, los árboles suelen tener una estructura ramificada (de hecho, de ahí viene su nombre), pudiéndose distinguir entonces entre distintas **ramas** del árbol. Siguiendo con la analogía, los puntos de la red afectados por el árbol que estén más alejados de la raíz del mismo pueden denominarse **hojas** del árbol.

El concepto de árbol de congestión está reflejado en la figura 2.7, que representa la misma red que los ejemplos anteriores en un momento en que la congestión se ha propagado desde el conmutador C hasta algunos *buffers* de los conmutadores A y B, formándose así las típicas series de *buffers* bloqueados "en cadena" que constituyen los árboles de congestión.

Realmente, la congestión puede originarse y expandirse en forma de árboles de congestión de muy diversos modos, dependiendo del tipo de tráfico involucrado en su generación y, también, de la arquitectura de los conmutadores de la red. Por ejemplo, si los conmutadores de la red representada en la figura 2.7 tuvieran un *speedup* de valor 2, los paquetes, en el conmutador C, se acumularían en el *buffer* del puerto de salida por el que existe contención antes que en los *buffers* de los puertos de entrada.

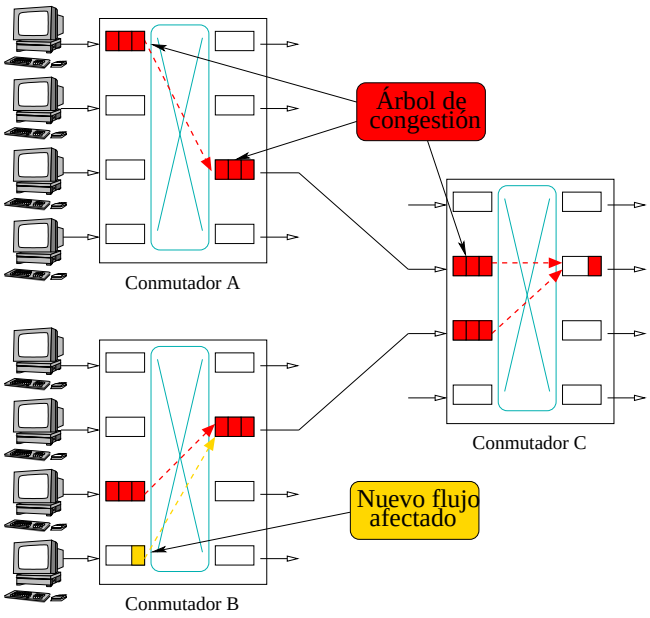


Figura 2.7: Formación de árboles de congestión.

Como veremos en un capítulo posterior, esta variedad en la aparición y propagación de la congestión no debería ignorarse por ninguna técnica de control de congestión que pretenda ser eficiente en cualquier entorno.

En la misma figura 2.7 puede apreciarse otro grave inconveniente de las situaciones de congestión: ésta afectará también a aquellos flujos de datos que no son causantes directos de la congestión. Basta con que un paquete solicite cruzar hacia un *buffer* que forma parte de un árbol de congestión para que dicho paquete se bloquee considerablemente. Nótese que esto es así independientemente de la intensidad del flujo de datos al que pertenece el paquete, y también independientemente de si sigue o no una ruta que pase por el punto de origen de la congestión (la raíz del árbol). En consecuencia, nos encontramos ante un problema que no afecta sólo a los flujos de datos que lo originan, sino a todo el tráfico presente en la red. Este hecho se concreta en un efecto bastante estudiado, el conocido como **bloqueo de cabeza de línea**, del que se ocupa el punto siguiente.

2.6.3. Bloqueo de cabeza de línea (*Head-Of-Line blocking*)

El fenómeno del bloqueo de cabeza de línea (o *Head-Of-Line (HOL) blocking*) sucede cuando paquetes situados a la cabeza de sus respectivos *buffers FIFO* se bloquean (debido, por ejemplo, a la contención por un puerto de salida), impidiendo que aquellos paquetes que lleguen al conmutador tras los bloqueados avancen hacia los puertos de

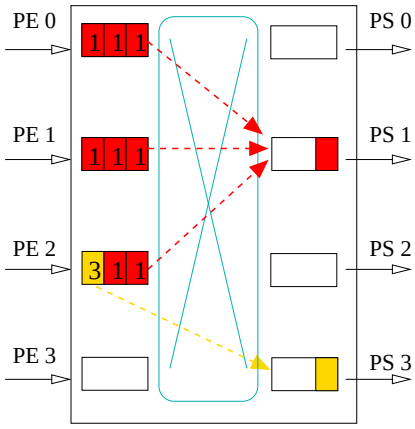


Figura 2.8: Efecto de bloqueo de cabeza de línea (*Head-Of-Line blocking*) en el *buffer* de un puerto de entrada.

salida a los que están destinados, independientemente de si dichos puertos de salida se encuentran libres o no en ese momento.

La figura 2.8 muestra un ejemplo de *HOL blocking* en un conmutador. Para mayor claridad, la figura muestra en cada paquete situado en los puertos de entrada el número del puerto de salida solicitado por dicho paquete. Como puede apreciarse, existe una fuerte contención por el puerto de salida 1 (*PS1*). Esto provoca que sólo uno de los paquetes situados en cabeza de los *buffers* de los puertos de entrada *PE0*, *PE1*, *PE2* podrá cruzar hacia *PS1*, bloqueándose el resto. Los paquetes bloqueados en cabeza bloquearán a su vez a otros paquetes almacenados en el *buffer*. Por ejemplo, el bloqueo del paquete situado a la cabeza del *buffer* del puerto *PE2* provoca el bloqueo del resto de paquetes de dicho *buffer*, incluyendo el de un paquete que solicita el puerto de salida *PS3* que, como puede verse, se encuentra libre.

Aunque el bloqueo de los paquetes situados en cabeza de los *buffers* tendrá una duración limitada (salvo algún tipo de error en el sistema), el *HOL blocking* puede ralentizar notablemente el avance de paquetes que, de no existir éste, no tendrían mayor problema para cruzar hacia los puertos de salida solicitados. En el ejemplo mostrado en la figura 2.8 no existe contención por el puerto de salida *PS3*, pero el paquete que lo solicita tendrá que esperar a que crucen dos paquetes que, al estar involucrados en una situación de contención por el puerto *PS1*, provocarán que dicha espera se prolongue durante una gran fracción del tiempo.

El *HOL blocking* es un serio problema para redes basadas en conmutadores de tipo *IQ* y *CIOQ*. De hecho, se calcula que el *HOL blocking* limita la eficiencia de un conmutador *IQ* a un 58 % del máximo posible [KHM87].

Además, el *HOL blocking* es un fenómeno directamente ligado a la congestión, ya que ésta puede provocar el bloqueo de los paquetes en cabeza de los *buffers* en todos

los conmutadores por donde crezcan los árboles de congestión. Cuanto más intensa sea la congestión, mayor será la probabilidad de que se produzca *HOL blocking* y mayor será el retardo introducido en los paquetes afectados. Además, cuanto más extensos sean los árboles de congestión, mayor será el conjunto de puntos de la red donde el *HOL blocking* pueda producirse.

Más aún, cabe plantearse si el *HOL blocking* no es la principal causa de la degradación de prestaciones (descenso de productividad, aumento de la latencia) que experimenta la red en situaciones de congestión. Esta cuestión tiene sentido si se considera que los paquetes pertenecientes a los flujos de datos que provocan la congestión sufrirán retardos inevitablemente, pues todos ellos deben cruzar el punto de origen de la congestión: la raíz del árbol. En cambio, los paquetes de flujos de datos afectados por la congestión, sin ser responsables de la misma, sufrirán debido al *HOL blocking* grandes retardos aun cuando en su camino no se encuentre la raíz del árbol. Esta observación, que es clave para nuestro trabajo, será retomada y analizada más ampliamente en el siguiente capítulo.

2.7. La especificación *PCI-Express Advanced Switching*

Como ejemplo de una tecnología de red de interconexión concreta, expondremos a continuación las características básicas de una de las tecnologías más recientes: *PCI-Express Advanced Switching* (en adelante, simplemente *Advanced Switching* ó *AS*). Es importante destacar que algunos aspectos de *Advanced Switching* presentados en esta sección aparecerán de nuevo en capítulos posteriores. Esto es debido a que nuestra propuesta de técnica de control de congestión plantea ciertos requisitos mínimos para su funcionamiento, y, al satisfacer *Advanced Switching* plenamente dichos requisitos, esta tecnología ha sido considerada siempre como la más inmediata para la implementación real de nuestra propuesta. Por otra parte, esto no significa en absoluto que la técnica propuesta en la presente tesis sea aplicable exclusivamente a *Advanced Switching*.

2.7.1. Características generales

Advanced Switching es un estándar abierto desarrollado por el *Advanced Switching Interconnect Special Interest Group* (*ASI-SIG*) [Gro], un grupo de trabajo impulsado desde varias empresas líderes del mundo de la computación y de la comunicación. La progresiva convergencia de ambos entornos ha acabado propiciando el mutuo interés por la existencia de un estándar de interconexión común, válido para la construcción de una amplia gama de sistemas (desde encaminadores IP hasta estaciones base de telefonía móvil) y que al mismo tiempo ofreciera las altas prestaciones demandadas a

las actuales redes de interconexión de alta velocidad. Un estándar de este tipo ahorraría a los fabricantes tiempo y dinero al sustituir dicho estándar a tecnologías diseñadas específicamente para sus sistemas.

Otra de las características deseables en dicho estándar sería que mantuviera la compatibilidad con sistemas anteriores. Como resultado, *Advanced Switching* está basado en la tecnología *PCI-Express* [Spe], a la que añade los protocolos necesarios para soportar mecanismos propios de las modernas tecnologías de interconexión (encapsulado de protocolos de orden superior, control de flujo, garantías de calidad de servicio, tráfico multidestino, etc.) [ASIb]. Esta filosofía le permite ser compatible con los numerosos sistemas y aplicaciones existentes basados en el extendido bus *PCI*, y al mismo tiempo ofrecer las prestaciones requeridas por sistemas más exigentes.

Hacia el final de 2003, se publicó la versión 1.0 de la especificación *Advanced Switching* [ASIa]. Aunque en muchos aspectos la especificación es flexible y permite bastante libertad de implementación, existen ciertas características generales básicas. Entre las más importantes cabe citar las siguientes:

- Empleo de enlaces punto a punto *full-duplex*. Esta es una característica "heredada" de la tecnología *PCI-Express*, que ya incorporaba este tipo de enlaces frente al modelo basado en medio compartido del bus *PCI* tradicional. El ancho de banda básico ($\times 1$) de estos enlaces es de 2,5 *Gbps* en cada sentido, escalable hasta 80 *Gbps* ($\times 32$) mediante el agregado en paralelo de varios enlaces serie (*multiple lanes*).

- Conmutadores sin restricciones en cuanto al número de puertos, y con una anchura variable del *lane* en los mismos.

- Libertad para la construcción de topologías de cualquier tipo (mallas, redes multietapa, irregulares, etc.) y de cualquier tamaño, desde 2 a miles de nodos. Esta flexibilidad también hace posible la existencia de caminos redundantes para mejorar la tolerancia a fallos en la red.

- Control de flujo basado en créditos.

- Conmutación de tipo *virtual cut-through*.

- Se permite el envío de mensajes tanto a un único destino (*unicast*) como a varios (*multicast*). En el primer caso, se emplea encaminamiento fuente.

- Los mensajes pueden encapsular cualquier protocolo de orden superior, lo que facilita la comunicación entre componentes de forma transparente. Además, se mantiene la compatibilidad con el modelo de *software* empleado en el bus *PCI* tradicional.

- Control de errores en la transmisión mediante el añadido de un número de secuencia y un *CRC* a los mensajes.
- Definición de distintas clases de tráfico (hasta 8) y canales virtuales (*virtual channels*). En conjunto, estos mecanismos permiten dar soporte a estrategias para la garantía de calidad de servicio (*QoS*).
- Establecimiento de diferentes mecanismos que pueden emplearse para el control de congestión (posibilidad de notificación a las fuentes para la limitación de inyección, control de flujo basado en ocupación del *buffer*, etc.).

Como ya se ha mencionado, algunas de estas características de *Advanced Switching* son especialmente relevantes para nuestra propuesta. En concreto, y como se justificará en un capítulo posterior, en nuestra técnica de control de congestión se asume el uso de un tipo de encaminamiento similar al empleado en *Advanced Switching*. Además, y como es lógico, resulta especialmente importante lo establecido en la especificación *Advanced Switching* respecto al control de congestión. Por tanto, nos ocuparemos con mayor detalle de estos dos aspectos en los siguientes puntos.

2.7.2. Encaminamiento en *Advanced Switching*

Como ya se ha indicado, *Advanced Switching* emplea encaminamiento fuente. Para implementarlo, un campo de la cabecera de los paquetes *Advanced Switching* se dedica a contener la información codificada que indica la ruta completa hasta el destino. Este campo consta de 31 bits, y es conocido como ***turnpool***. El *turnpool* se divide en varios fragmentos, cada uno destinado a ser "leído" y decodificado por uno concreto de los conmutadores que debe cruzar consecutivamente un paquete en su ruta. Cada fragmento del *turnpool* establece el desplazamiento que debe realizar el paquete entre su puerto de entrada y un puerto de salida en el conmutador correspondiente. Este método es conocido como **direccionamiento relativo al puerto de entrada**.

Para indicar a un conmutador en qué posición del *turnpool* se encuentra el fragmento que le corresponde leer, la cabecera de los paquetes *Advanced Switching* contiene un puntero (***turn pointer***) que indica el bit del *turnpool* donde debe comenzar la lectura el siguiente conmutador de la ruta. Por tanto, el valor del *turn pointer* debe actualizarse una vez realizado el cruce de un paquete en un conmutador. En la figura 2.9 puede apreciarse la posición ocupada tanto por el *turnpool* como por el *turn pointer* dentro de una cabecera de paquete *Advanced Switching*.

Como puede apreciarse en la figura 2.9, la cabecera incluye además un **bit de dirección**, que indica si el desplazamiento del paquete debe realizarse en sentido positivo o negativo respecto al puerto de entrada. Dependiendo del valor de este bit, el conmutador calcula el puerto de salida del paquete sumando (bit de dirección a 0) o restando

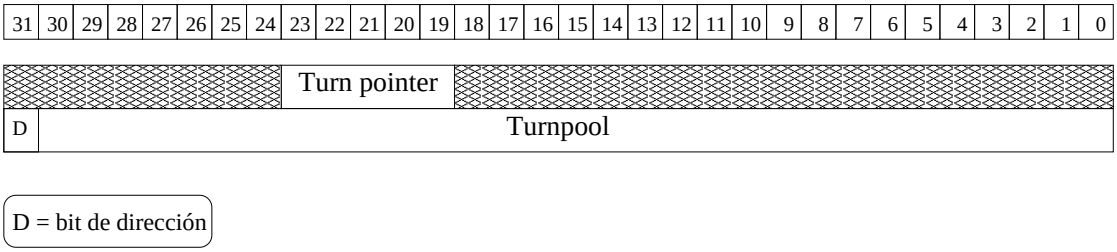


Figura 2.9: Posición del *turnpool* y el *turn pointer* dentro de la cabecera de un paquete *Advanced Switching*.

(bit de dirección a 1) al identificador del puerto de entrada del paquete el valor del desplazamiento obtenido de la lectura y decodificación del fragmento correspondiente del *turnpool*.

La figura 2.10 muestra un ejemplo de cómo se produce el encaminamiento de paquetes a través de la red. Cuando el paquete llega al conmutador 1 (punto A), éste lee, en la posición indicada por el *turn pointer*, su fragmento correspondiente del *turnpool*, y lo decodifica[1] para obtener el valor del desplazamiento (4) que debe realizar el paquete. Puesto que el bit de dirección de la cabecera está a 0, el valor del desplazamiento se suma al del identificador del puerto de entrada del paquete (puerto 3) para calcular el puerto de salida al que debe encaminarse dicho paquete (puerto 7). Nótese que una vez que el paquete ha cruzado el conmutador 1, (puntos B y C) el *turn pointer* de la cabecera se ha actualizado para "apuntar" al desplazamiento que corresponde al siguiente conmutador de la ruta (conmutador 2). Éste, a su vez, repetirá los mismos pasos que el conmutador 1, de modo que el paquete llegará hasta el punto D y seguirá avanzando hasta que llegue a su destino.

Nótese que con este mecanismo de encaminamiento es posible identificar una parte de un camino completo dentro de la red (o sea, una ruta que no comience o acabe necesariamente en un terminal), considerando sólo un subconjunto de desplazamientos consecutivos de entre todos los indicados en un *turnpool*. Por tanto, es posible determinar desde cualquier conmutador de la red si un paquete pasará o no más adelante por un punto concreto de la red. Por ejemplo, en el caso mostrado en la figura 2.10, puede saberse desde el punto A que el paquete pasará por el punto D. Conviene tener presente esta observación, pues resulta fundamental, como veremos, para el modo en que nuestra propuesta identifica los posibles puntos congestionados en la red.

[1]Para mayor claridad, la figura muestra directamente el valor decodificado de los desplazamientos en el *turnpool*.

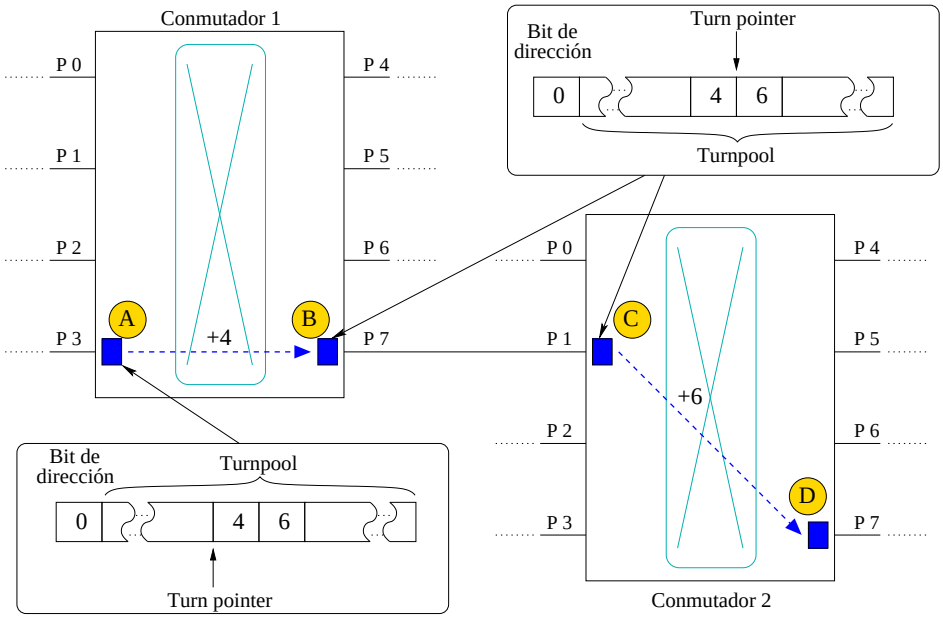


Figura 2.10: Avance de un paquete *AS* a través de la red en función de los valores del *turnpool*, el *turn pointer* y el bit de dirección almacenados en su cabecera.

2.7.3. Control de congestión en *Advanced Switching*

Respecto al control de la congestión, el estándar *Advanced Switching* no define una estrategia precisa que deba implementarse obligatoriamente. En su lugar, se sugieren una serie de mecanismos opcionales que pueden usarse, individual o combinadamente, para implementar diferentes técnicas de control de congestión. Pero en cualquier caso, se deja en manos de los fabricantes la implementación o no de tales mecanismos, el ajuste de los mismos y su aplicación concreta para soportar cualesquiera estrategias de control de congestión.

A continuación se describen muy brevemente estos mecanismos opcionales:

- **Control de flujo basado en estado.** Se trata de un control de flujo especial del tipo "basado en marcas" (o sea, de tipo *Stop & Go*). En este caso, el estándar denomina a este control de flujo *Status-Based Flow Control*, pero también es conocido por el nombre que reciben tanto las dos marcas (niveles) de ocupación como sus correspondientes notificaciones: *Xon/Xoff* (donde *Xoff* equivale al nivel de *Stop* y *Xon* al nivel de *Go*).
- **Limitación de inyección en los terminales.** Este mecanismo propone notificar de algún modo la existencia de congestión a los terminales para que ajusten (reduzcan) su tasa de inyección de paquetes en la red. El estándar sugiere que este ajuste se realice en función de las posibles divisiones del tráfico permitidas en *Advanced Switching* (clases de tráfico, canales virtuales, etc.).

- **Planificación para garantizar un mínimo ancho de banda en los enlaces de salida.** Este mecanismo se basa en garantizar a uno de los canales virtuales de los enlaces de salida máxima prioridad y cierta fracción del ancho de banda del enlace. El resto del ancho de banda disponible se reparte entre el resto de canales virtuales de ese enlace con arreglo a determinados criterios. El canal prioritario puede utilizarse para enviar mensajes que informen de eventos en la red (*Fabric Management Channel*), incluyendo la aparición de congestión. Así, por ejemplo, un mensaje que notifique congestión y que use este canal tendría un ancho de banda mínimo garantizado y la menor latencia posible, con lo que se podría reaccionar con mayor rapidez para solucionar la congestión. Por otra parte, se garantiza que estos mensajes no consumirán todo el ancho de banda disponible en los enlaces, al tener su ancho de banda limitado.

- **Descarte de paquetes.** En última instancia, *Advanced Switching* ofrece como solución a la congestión la opción de descartar los paquetes congestionados. Se sugiere además en este caso que los paquetes se marquen como "descartables" o "no descartables". Evidentemente, esta opción no es aceptable para tráficos con exigencias estrictas respecto a la recepción de paquetes.

Además de estos mecanismos opcionales establecidos en el estándar, se permite a los fabricantes el implementar mecanismos adicionales relativos al control de congestión. En la especificación se citan (a modo meramente "informativo") otros mecanismos adicionales, basados todos en *software*: control de admisión, selección de rutas no congestionadas y adaptación a la congestión (monitorización de la red para detectar congestión y reaccionar a ésta de algún modo).

En el siguiente capítulo nos centraremos en analizar las ventajas e inconvenientes de las diversas aproximaciones al control de congestión propuestas en la literatura. Como se verá, la mayoría de los mecanismos (opcionales o "informativos") de *Advanced Switching* expuestos en la presente sección corresponden a determinados enfoques previamente propuestos para solucionar este problema. Por tanto, explícita o implícitamente, volveremos sobre estos mecanismos, desde un punto de vista crítico y no meramente descriptivo, en algún punto del próximo capítulo.

Capítulo 3

Control de congestión

Una vez explicados los conceptos elementales relativos al entorno donde se sitúa la técnica propuesta en la presente tesis, y habiendo caracterizado el problema que con ella pretendemos solucionar, es momento de revisar otras soluciones que se han propuesto y aplicado anteriormente para el problema de la congestión, y también de establecer la filosofía básica de nuestra propuesta frente a las diversas tendencias seguidas por otras técnicas.

En primer lugar analizaremos por qué hoy en día, teniendo en cuenta las características de las tecnologías empleadas en las redes de interconexión actuales, es realmente necesario emplear técnicas específicas de control de congestión.

A continuación, nos ocuparemos de aquellas técnicas concretas desarrolladas según ciertos enfoques que podríamos considerar "tradicionales". Básicamente, estos enfoques se orientan a la evitación o la eliminación de la congestión en sí.

Seguidamente, expondremos cuál es el enfoque básico de nuestra propuesta y las razones que nos llevan a considerarlo más apropiado que los tradicionales. En concreto, y como se justificará más adelante, nuestra técnica aborda el problema de la congestión tratando de eliminar su principal efecto negativo (el *HOL blocking*, tal y como hemos apuntado en el capítulo anterior) en lugar de la propia congestión.

Por último, se analizarán las ventajas e inconvenientes de aquellas técnicas de control de congestión anteriores que han adoptado un enfoque similar respecto al problema de la congestión. Como veremos, ninguna de ellas puede considerarse a la vez realmente eficiente y escalable, lo que da sentido a nuestros esfuerzos por conseguir desarrollar una técnica de tales características.

3.1. Importancia actual de las técnicas de control de congestión

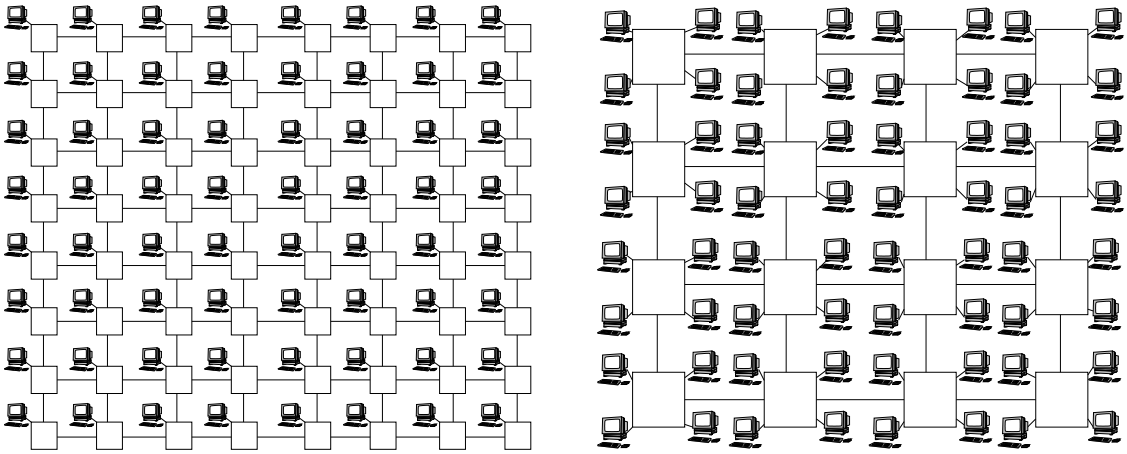
Llegados a este punto, y teniendo en cuenta que pretendemos desarrollar una técnica de control de congestión, es imprescindible realizarse la siguiente pregunta: ¿Realmente es necesario emplear técnicas de control de congestión en las redes de interconexión actuales?. O, de otro modo: ¿Realmente ocurrirán situaciones de congestión en los sistemas que emplean estas redes?.

Estas cuestiones no carecen en absoluto de sentido, sobre todo teniendo en cuenta que hasta hace relativamente pocos años, la congestión no era considerada un problema. Esta postura se justificaba por la práctica, habitual tiempo atrás, de **sobredimensionar** la red. En una red sobredimensionada, se emplean muchos más componentes de los estrictamente necesarios para interconectar los terminales, de manera que las prestaciones teóricamente ofrecidas por la red sean muy superiores a las requeridas por el sistema. En estas configuraciones, la utilización media de los enlaces será baja, lo que reducirá la contención y, en consecuencia, la probabilidad de situaciones de congestión.

Como ejemplo, la figura 3.1 muestra dos topologías posibles (en ambos casos de tipo malla) para interconectar 64 terminales. En el sistema representado en la figura 3.1.a (sistema A) se ha conectado un único terminal a cada conmutador, lo que hace necesario usar 64 conmutadores. En cambio, la figura 3.1.b muestra un sistema (sistema B) en el que se han conectado 4 terminales por conmutador, bastando en este caso 16 conmutadores para construir la red. En ambos sistemas, la red de interconexión es totalmente conexa, y por tanto es válida para permitir la comunicación entre todos los terminales. Nótese además que si el número de terminales que son nodos de procesamiento es el mismo en ambos casos, la capacidad de cómputo de los dos sistemas será idéntica. Pero puede intuirse que, en igualdad de condiciones respecto al tráfico inyectado por los terminales, la contención por conseguir el uso de los enlaces será mucho menor en el caso del sistema A.

Para confirmar esta intuición, la figura 3.2 muestra, de forma aproximada, las prestaciones ofrecidas por las redes de los sistemas representados en la figura 3.1. Se ha recurrido a una de las formas más habituales para reflejar gráficamente las prestaciones de la red de interconexión: la representación de las curvas de latencia en función del tráfico inyectado por los terminales[1]. En este tipo de gráficas, el punto de saturación indica cuál es el máximo tráfico inyectado por los nodos de procesamiento que soporta la red. Más allá de este punto el valor de la latencia aumenta drásticamente, lo que indica que la capacidad de los enlaces se ha visto desbordada por el tráfico (se dice

[1] La gráfica resultante para una determinada red dependerá de muchos factores, como la topología, el algoritmo de encaminamiento, el patrón de destinos de los mensajes, tamaños de mensajes, etc.



(a) Sistema A (b) Sistema B

Figura 3.1: Ejemplo de dos sistemas. (a) 64 terminales y 64 conmutadores y (b) 64 terminales y 16 conmutadores.

que la red está "saturada"). Evidentemente, en la zona de saturación, la probabilidad de que exista contención y congestión es mucho mayor.

Como puede verse en la figura 3.2, a igualdad de condiciones (mismo ancho de banda de los enlaces, mismo número total de terminales en el sistema), a menor número de terminales por conmutador en la red, mayor carga de tráfico se requiere para llegar al punto de saturación. Así, la red del sistema A satura para una carga de tráfico mayor que la carga para la que satura el sistema B. Ambos sistemas funcionarán sin problemas mientras no se superen los respectivos puntos de saturación. Por tanto, si el tráfico inyectado por los terminales en ambos sistemas se sitúa dentro de la zona de trabajo mostrada en la figura 3.2, ninguno de los dos sistemas trabajará en la zona de saturación. Las prestaciones de la red del sistema B se ajustan estrechamente a la

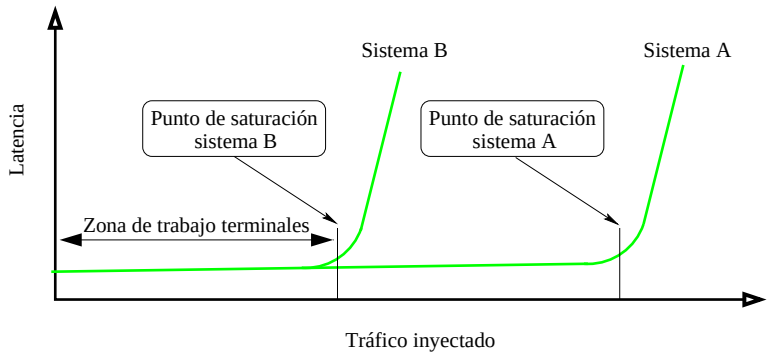


Figura 3.2: Representación aproximada de la latencia de red frente al tráfico inyectado para los sistemas de la figura 3.1.

zona de trabajo de los terminales. En el caso del sistema A, las prestaciones de la red superan con mucho a las requeridas y, en principio, no se aprovecharían al máximo, por lo que la red se considera sobredimensionada.

Ahora bien, el tráfico inyectado en redes reales no suele ser uniforme, ni en la distribución de destinos de los paquetes, ni en la tasa de generación de los mismos desde un mismo nodo. Generalmente, existirán ráfagas intensas de tráfico, quizá con alta localidad en cuanto al destino, que harán que aumente repentinamente el tráfico inyectado. Por tanto, en redes no sobredimensionadas, es muy probable que en estas ocasiones se rebase el punto de saturación, comprometiendo las prestaciones de la red. En cambio, las redes sobredimensionadas serán menos sensibles a estas ráfagas de tráfico al ser mayor su punto de saturación. Dicho de otro modo, en redes sobredimensionadas será más difícil que se llegue a situaciones donde la probabilidad de congestión aumente.

Teniendo en cuenta lo anteriormente expuesto, podría pensarse que sobredimensionar la red es una solución sencilla para el problema de la congestión, y que por tanto las técnicas específicas de control de congestión son innecesarias. De hecho, este razonamiento pudo ser válido hace algún tiempo. Actualmente, sin embargo, las tecnologías de redes de interconexión presentan ciertas características que dificultan enormemente su sobredimensionamiento. Concretamente, hoy en día existen dos importantes razones para reducir en lo posible el tamaño de una red:

- **El aumento del coste de los componentes** en relación con el coste de los procesadores. Aunque la popularidad de las modernas tecnologías de redes de interconexión es cada día mayor, ninguna de las empresas fabricantes mantiene hoy en día un nivel de ventas lo suficientemente grande como para poder abaratar el precio de los componentes y al mismo tiempo compensar los gastos de diseño y producción. Por el contrario, el volumen de ventas de los grandes productores de procesadores les permite ofrecer productos asequibles, salvo excepciones muy concretas. En consecuencia, los enlaces y conmutadores de redes comerciales de altas prestaciones (*Myrinet*, *Infiniband*, *Quadrics* [QsN], etc.) resultan caros en comparación con los procesadores empleados en casi cualquier sistema.

- **El mayor consumo de potencia de los componentes** (enlaces y conmutadores) de las redes de interconexión. Este aumento del consumo se debe a la evolución de la tecnología. Por una parte, el incremento de la velocidad de transmisión de los enlaces y de la velocidad de conmutación en los *crossbars* ha elevado las frecuencias de las señales implicadas, aumentando por tanto la fracción del consumo total del sistema que se dedica a la red de interconexión (hasta el 58 % en la circuitería de interconexión integrada en el procesador *Alpha* 21364 [MBL02]). Por otra parte, en las tecnologías modernas, el consumo de potencia en los enlaces es prácticamente independiente de su grado de utilización para la transmisión de tráfico convencional, debido a la abundante señalización y mensajería de control

necesarias para soportar determinados mecanismos. En muchas tecnologías, por ejemplo, los puertos activos envían por el enlace al que se conectan señales "de latido" para informar de este estado de actividad al puerto situado en el otro extremo del enlace. Incluso en ciertas tecnologías se repiten periódicamente procesos de "exploración" de la red mediante mensajes de control específicos que recorren la topología completamente [BCF95].

En cuanto al consumo de potencia, se han propuesto ciertas técnicas [SPJ03] que tratan de reducir el consumo de los enlaces mediante un ajuste dinámico de su ancho de banda en función de su utilización. En situaciones de baja utilización del enlace, se reduciría su frecuencia (y voltaje) de funcionamiento, disminuyendo por tanto el consumo de potencia. Genéricamente, estos mecanismos reciben el nombre de técnicas de escalado dinámico del voltaje (*dynamic voltage scaling, DVS*), y, además de para enlaces, se han propuesto para otros dispositivos, como microprocesadores [BB00]. Sin embargo, el empleo de estas técnicas en enlaces presenta diversos problemas. En primer lugar, es necesario que la tecnología de los enlaces permita el ajuste dinámico de la frecuencia. Por otra parte, el ajuste de la frecuencia debe realizarse con arreglo a un historial de uso del enlace, lo que dificulta y ralentiza la reacción del mecanismo ante variaciones repentinas del tráfico (ráfagas). Además, la eficiencia y reducción del consumo conseguidas por estas técnicas parecen ser menores que las alcanzadas por mecanismos de escalado estático del voltaje combinados con el uso de encaminamiento adaptativo [SC04]. Y, en cualquier caso, las técnicas basadas en *DVS* no resuelven en ningún modo el problema del excesivo coste de las redes sobredimensionadas.

En definitiva, no parece viable en las actuales circunstancias sobredimensionar la red. Por tanto, la red de interconexión de la mayoría de sistemas no estará sobredimensionada, y será habitual que varios terminales se conecten a un mismo conmutador. Las tecnologías actuales de redes de interconexión permiten una gran flexibilidad en cuanto a la conectividad y, desde este punto de vista, no hay mayor problema. Pero desde el punto de vista operativo, es necesario asumir que el tráfico inyectado se situará cerca del punto de saturación de la red, existiendo una probabilidad considerable de aparición de situaciones de congestión.

Es posible utilizar algunas técnicas que, aun sin ocuparse expresamente de la congestión, retrasan su aparición o alivian ligeramente su intensidad, como pueden ser el encaminamiento adaptativo [Dua93, KS91, GY93, TLM03, SDT04] o las técnicas de balanceo de carga [FGL99, SDT04]. Pero en cualquier caso ninguna de ellas puede evitar la degradación de prestaciones que experimenta la red cuando se llega a situaciones de congestión.

Por tanto, es necesario tomar medidas eficaces que se ocupen específicamente de la congestión que pueda aparecer en la red. Podría considerarse como solución a la congestión el descarte de los paquetes congestionados, lo que eliminaría radicalmente

el problema. Esta es la solución adoptada en los sistemas o redes "con pérdidas", que generalmente son redes de comunicaciones, como *Internet*. Pero esta práctica puede introducir grandes retardos debido a las retransmisiones de los paquetes descartados, y por tanto no es aceptable en los entornos donde suelen emplearse la mayoría de las modernas redes de interconexión (como por ejemplo, sistemas de computación de altas prestaciones), que son redes "sin pérdidas". Cabe recordar que *Advanced Switching* sí que contempla esta posibilidad, pero sólo como una opción, en la línea del espíritu "abierto" de este estándar.

En conclusión, dadas las características tanto de las modernas tecnologías de redes de interconexión como de los sistemas de los que forman parte, es imprescindible utilizar algún mecanismo específico que controle de algún modo las situaciones de congestión para mantener en la medida de lo posible las prestaciones de la red. Con este propósito se han propuesto diversas técnicas de control de congestión, que analizaremos en los puntos siguientes.

3.2. Enfoques tradicionales del control de congestión

El diseño de técnicas de control de congestión para redes sin pérdidas ha atraído el interés de los investigadores desde hace tiempo [PN85]. Debido a los distintos puntos de vista desde los que se ha abordado el problema de la congestión, han ido surgiendo diversos enfoques o filosofías de diseño claramente diferenciados.

En general, los primeros esfuerzos se centraron en conseguir la eliminación de la congestión en sí, intentando que las situaciones de congestión no pudieran existir en la red. Aunque las técnicas diseñadas con este enfoque "tradicional" se pueden clasificar con arreglo a distintos criterios [Dan99, YR95], generalmente se distinguen dos tendencias básicas: las estrategias proactivas y las reactivas, que se explican seguidamente.

3.2.1. Técnicas proactivas

Aquellas técnicas que podemos denominar proactivas tienen en común el intentar que la congestión no llegue a producirse en la red bajo ninguna circunstancia, ni siquiera durante un tiempo limitado. En ocasiones, estas técnicas reciben también la calificación de "técnicas de evitación" de la congestión, y, en general, requieren una "planificación" previa a la actividad de la red.

Algunas de estas técnicas intentan evitar que se creen puntos de congestión en la red (*hot-spots*). Por ejemplo, en un sistema de almacenamiento es posible distribuir los datos en distintos dispositivos, o proporcionar distintos caminos para llegar a ellos, de modo que se minimice la contención que pudiera darse en el acceso a dichos datos.

Existen implementaciones de este tipo de técnicas mediante *software* [YTL87, BLK93] y mediante *hardware* [WSN95]. El principal inconveniente de estas técnicas es que no siempre es posible su implementación, bien por estar diseñadas para entornos muy específicos (como multiprocesadores), bien por requerir la existencia de componentes adicionales en la red (lo cual, como hemos visto anteriormente, no es viable actualmente).

En este grupo podrían incluirse también aquellas técnicas que realizan una planificación previa de cada transmisión, lo que asegura, entre otras cosas, que no existirán árboles de congestión. Esta planificación se basa en un conocimiento "a priori" de los recursos necesarios para cada transmisión, y en la reserva previa de dichos recursos para garantizar que el flujo de datos discurra sin problemas entre origen y destino. El problema es que los datos necesarios para planificar la transmisión no siempre están disponibles en todos los posibles entornos. Además, la planificación y la reserva de recursos para cada transmisión introducen retardos, y se complican en redes donde los paquetes tengan que cruzar en sus rutas por un número medio o alto de conmutadores. En general, este tipo de técnicas son realmente propias de entornos orientados a ofrecer garantías de calidad de servicio (como es el caso de *ATM* [dP95]), especialmente a flujos de datos multimedia. Por tanto, estas técnicas no son demasiado adecuadas para entornos de computación de altas prestaciones, donde los flujos típicos de datos son breves en comparación con la larga permanencia en la red de los flujos multimedia.

3.2.2. Técnicas reactivas

A diferencia de las técnicas proactivas, las técnicas reactivas sí permiten que sucedan situaciones de congestión, pero proponen algún tipo de mecanismo que de algún modo detecta dichas situaciones y las elimina "a posteriori". En algunas ocasiones, estas técnicas se denominan "técnicas de detección" o, más comúnmente, "técnicas de recuperación" de la congestión. Generalmente se basan en un modelo de control en bucle cerrado, en el que tras la detección de la congestión, se notifica su existencia a las fuentes de tráfico para que adopten las medidas oportunas para solucionar la situación. Habitualmente, estas medidas implican reducir, o incluso detener, la inyección de paquetes; recordemos que éste es uno de los mecanismos opcionales que propone *Advanced Switching* para el control de congestión.

En cuanto al modo de detectar la congestión, existen diversas propuestas. Por ejemplo, es posible medir el tiempo de espera de los paquetes bloqueados [KLC94, KLC97, HA97, ST98], aunque en este caso hay que tener en cuenta que este tiempo dependerá de la longitud de los paquetes. Otra opción es monitorizar el nivel de ocupación en los *buffers* de los conmutadores [Vea00, LSC95, TLM01], o la cantidad de peticiones pendientes de acceso a memoria [SLS90]. En el caso de monitorizar la ocupación de los recursos, la información enviada puede ser local (ocupación de recursos en un punto

concreto de la red) o global (ocupación de todos los recursos en la red) [TLM01]. Evidentemente, la información global proporciona un mayor conocimiento sobre el estado de la red, pero resulta mucho más costosa de implementar, y además requiere la transmisión de gran cantidad de mensajes de control, que consumen una importante fracción del ancho de banda. Los mecanismos basados en información local son más limitados, pero su implementación es más asequible y no necesitan mensajería adicional.

Respecto a las notificaciones sobre la congestión, algunas técnicas plantean su envío a todas las fuentes de tráfico [Vea00, TLM01, ST98], en cuyo caso el mecanismo consumiría un excesivo ancho de banda y además penalizaría a todas las fuentes, estén involucradas o no en la congestión. También es posible notificar congestión sólo a aquellas fuentes que envíen paquetes hacia la zona congestionada [KLC97], lo que permite una mayor eficiencia. Otras propuestas de técnicas reactivas [DA93, LD93, BLD01, BL03] notifican la congestión sólo a aquellas fuentes conectadas directamente al conmutador donde se detecta la congestión. Este último planteamiento presenta, a diferencia de otras técnicas, la ventaja de no necesitar la transmisión de información adicional, aunque penaliza en exceso a las fuentes locales, que no tienen por qué ser necesariamente las implicadas en la congestión. De hecho, estas técnicas consiguen buenos resultados con ciertos patrones de tráfico sintético, pero suelen fallar con patrones de tráfico realista.

El paso final del proceso de control mediante técnicas reactivas es la realización por parte de las fuentes de algún tipo de acción que resuelva la situación de congestión detectada. Aunque existen otras opciones [ST98], el método más ampliamente utilizado es, con diferencia, la limitación de inyección (*message throttling*) [DA93]. En este caso, tras recibir las notificaciones de congestión, las fuentes detienen, o al menos disminuyen, la inyección de paquetes en la red. Así, mientras algunas propuestas optan por detener completamente la inyección de mensajes al detectar congestión [LD93, PV96, LMD97, LMD98, ST98, TLM01, BLD00, BLD01], otras plantean reducir progresivamente el ancho de banda disponible para inyectar nuevos mensajes. Esta reducción puede realizarse bien disminuyendo el número de canales de inyección [BLD02], bien insertando intervalos de espera entre las transmisiones de mensajes [KLC94, KLC97]. Por otra parte, la limitación de la inyección puede aplicarse durante intervalos predefinidos de tiempo [ST98, KLC94, KLC97], o sin límites precisos, esperando por ejemplo hasta que el tráfico disminuya lo suficiente [LMD97, LMD98, BLD00, TLM01].

Sin embargo, la filosofía de control en bucle cerrado de las técnicas reactivas plantea un grave problema relativo al tiempo de reacción del mecanismo ante la congestión. Hay que tener en cuenta que la duración del proceso de control comprende un tiempo para detectar la congestión y, además y principalmente, un tiempo para que las notificaciones de congestión lleguen a las fuentes. Este último tiempo depende de la velocidad de los enlaces, pero también de la longitud de las rutas que tengan que recorrer las notificaciones. La figura 3.3 muestra una notificación que debe pasar por 3 conmutadores hasta llegar a la fuente notificada. Lógicamente, si se aumenta el tamaño

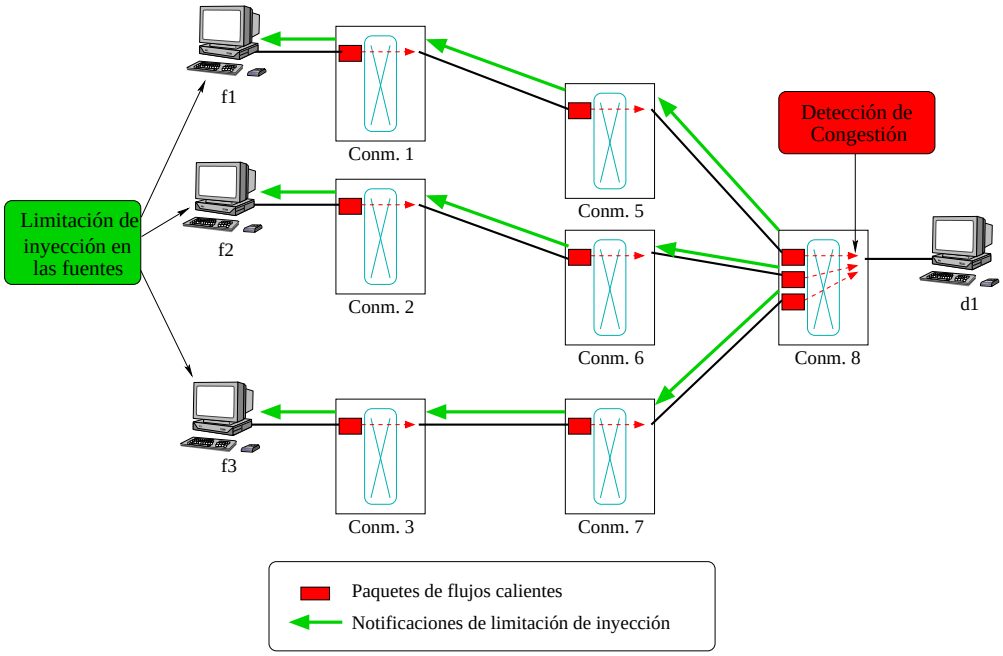


Figura 3.3: Proceso de detección y notificación de la congestión en un sistema de control en bucle cerrado con limitación de inyección.

de la red, la longitud de la ruta podría ser mayor, y más tiempo tardaría la notificación en llegar a la fuente. Esto implica que entre la detección de la congestión y la llegada de las notificaciones puede transcurrir demasiado tiempo como para que la reacción de las fuentes, sea cual sea, consiga eliminar de forma eficaz situaciones de congestión que quizá ya sean irreversibles.

Más aún, en el caso de que la solución a la congestión pase por limitar la inyección, hay que considerar que durante el tiempo que tarde en llegar la notificación a la fuente, ésta seguirá inyectando a la red tráfico que puede contribuir a la congestión (como muestra la figura 3.3). Téngase en cuenta que debido al uso casi universal en las modernas redes de interconexión de enlaces segmentados con gran ancho de banda, para cuando se tomen medidas, la cantidad de información "en vuelo" puede ser considerable. Y, de nuevo, a mayor tamaño de la red, mayor cantidad de enlaces compondrán las rutas recorridas por las notificaciones y mayor cantidad de paquetes se encontrarán viajando por la red.

Incluso es posible que la lentitud en la respuesta de estas técnicas dé lugar a oscilaciones e inestabilidades en la red, ya que puede que las fuentes reaccionen a la notificación aplicando alguna medida para solucionar una situación de congestión que, para entonces, ya haya desaparecido.

En definitiva, la respuesta a la congestión de las técnicas reactivas será lenta y posiblemente producirá inestabilidades en la red, lo que mermará notablemente su

eficacia. Nótese además que se trata de técnicas no escalables, ya que la eficacia de la respuesta es inversamente proporcional al tamaño de la red y al ancho de banda de los enlaces.

3.3. Enfoque alternativo para el control de congestión

Como hemos comprobado en las secciones anteriores, las soluciones "clásicas" para el control de congestión no parecen adecuarse a las características de las actuales redes de interconexión, ni tampoco a las de los entornos de altas prestaciones donde éstas suelen emplearse.

Una característica común a todas las técnicas previamente expuestas es que tratan de solucionar el problema de la congestión intentando que ésta desaparezca de un modo u otro. En esta sección analizamos un enfoque alternativo, basado no en la evitación o eliminación de la congestión en sí, sino en la reducción o eliminación de sus efectos negativos. Esta filosofía, por otra parte, es la que sigue la técnica de control de congestión propuesta en la presente tesis.

3.3.1. Ideas básicas

El enfoque de las técnicas tradicionales, basado en impedir de algún modo la presencia de congestión en la red, puede parecer en principio el más natural, e incluso el único posible, para solucionar el problema de la congestión. Sin embargo, este enfoque no tiene en cuenta un hecho que a nuestro juicio es fundamental: la existencia de puntos congestionados, y la de los correspondientes árboles de congestión, no es un problema en sí mismo.

Para explicar esta idea, recordemos que la congestión se forma cuando varios flujos de datos (a los que llamaremos "flujos calientes") confluyen en un punto de la red, produciéndose contención prolongada por la solicitud concurrente de ciertos recursos. Pero nótese que la red no puede atender mejor a estos flujos calientes: el enlace de salida del punto congestionado estará ocupado al 100 % de su capacidad, y no habrá modo de que los paquetes de estos flujos avancen más rápido. Por tanto, la productividad en ese enlace no puede ser mayor, y no se le puede "responsabilizar" de la bajada de productividad global que se produce en situaciones de congestión. Además, el control de flujo, que propaga la congestión en forma de árboles, garantiza que ningún *buffer* se desbordará, por muy intensos que sean los flujos calientes. Más aún, si el árbol de congestión acaba llegando hasta las fuentes de estos flujos, se producirá de forma implícita un efecto de limitación de inyección, en el que el propio control de flujo regulará perfectamente la tasa de envío de paquetes hacia el punto de congestión.

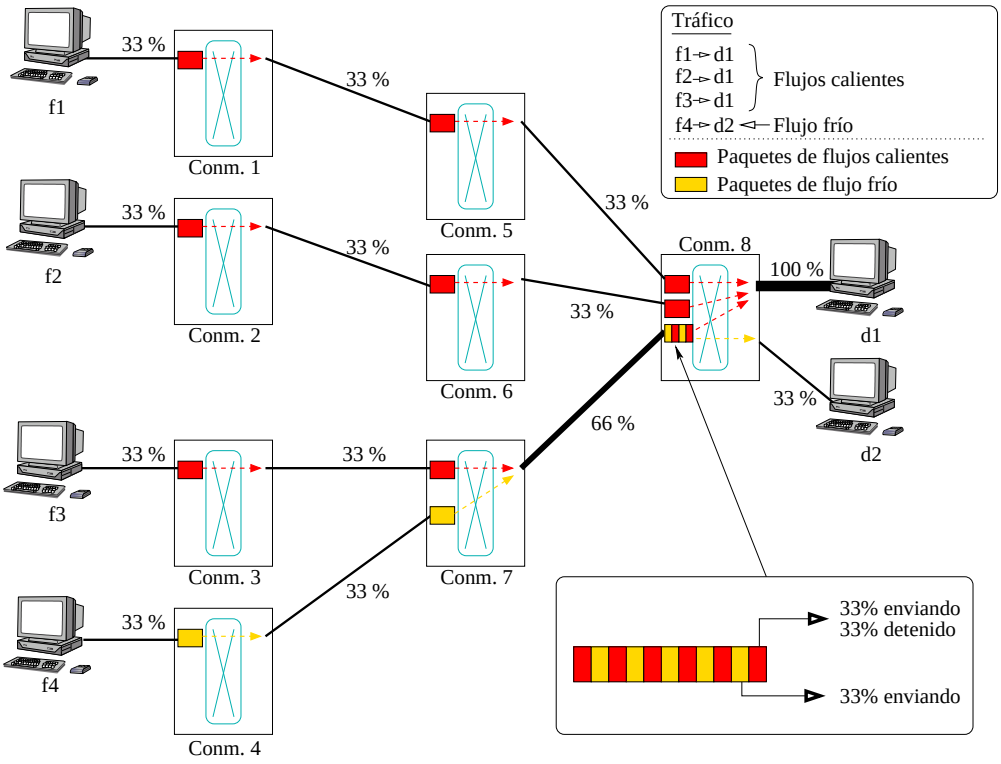


Figura 3.4: Ejemplo de la influencia de un árbol de congestión en un flujo de paquetes que no contribuye a la congestión.

Por consiguiente, parece que la mera presencia de paquetes congestionados en la red no explica por sí sola la degradación de prestaciones propia de situaciones de congestión. Así pues, cabe preguntarse qué circunstancias deben darse, además de la existencia de flujos calientes, para que se produzca esta degradación. La respuesta a esta cuestión ya se apuntó en el capítulo anterior: el principal efecto negativo de la congestión se produce en aquellos flujos de paquetes que no son responsables de la congestión, pero que acaban viéndose afectados por ella al coincidir parte de su ruta con algún tramo del árbol de congestión. En estos casos, los paquetes de estos flujos (a los que, por oposición, denominaremos "flujos fríos") se bloquearán en algunos puntos y avanzarán con una velocidad menor de la que permitirían los enlaces de salida a los que están destinados, que estarán funcionando por debajo de su capacidad. Es decir, que para que la congestión sea realmente nociva, deben existir "interferencias" entre los flujos calientes y los flujos fríos. Como ya se explicó en el tema 2, esta interferencia se concreta en el fenómeno del *HOL blocking*.

Para reforzar este planteamiento, la figura 3.4 muestra un ejemplo de cómo los flujos calientes influyen en el avance de los flujos fríos, y de cómo este hecho perjudica a las prestaciones globales de la red. En la figura, los enlaces se representan con distinto grosor según su utilización en el estado final del sistema (cuando el árbol de congestión se ha formado totalmente). Para mayor claridad, se muestran conmutadores *IQ*, pero

el ejemplo sería perfectamente válido para conmutadores *CIOQ* con valores de *speedup* normales (entre 1 y 2).

En el ejemplo mostrado en la figura, tres fuentes ($f1$, $f2$, $f3$) envían paquetes a un mismo destino ($d1$), lo que origina tres flujos de paquetes que confluyen en un puerto de salida del conmutador 8 (el que está conectado al destino). Cuando la inyección comienza, las fuentes pueden enviar paquetes a su máxima tasa de inyección (utilizan todo el ancho de banda disponible de los enlaces). Ahora bien, llegará un momento en que aparecerá contención entre estos flujos al solicitar la misma salida en el conmutador 8. Por el momento, las tres fuentes seguirán inyectando paquetes con la máxima tasa de inyección, y se irá formando un árbol de congestión al llenarse sucesivamente (desde la raíz hasta las fuentes) los *buffers* situados en las rutas de estos tres flujos. En el ejemplo, una vez el árbol está totalmente formado, las fuentes de los flujos calientes se convierten en las hojas del árbol, de modo que el control de flujo actuará limitando la inyección desde las fuentes. Por tanto, todos los paquetes de los flujos calientes avanzarán a un ritmo muy bajo, utilizando en este caso concreto $\frac{1}{3}$ del ancho de banda de los enlaces al ser tres los flujos que comparten un único enlace en la raíz del árbol. Pero, en lo que respecta a dicho enlace, conectado al destino, su utilización es la máxima (100 %), y por tanto la lentitud en el avance de los paquetes pertenecientes a flujos calientes no puede evitarse (a menos que se utilice un enlace con mayor ancho de banda). Es decir, si sólo existieran en la red los tres flujos calientes, la productividad de la red sería la máxima.

Sin embargo, en el ejemplo también existe un flujo frío que discurre entre la fuente $f4$ y el destino $d2$. Parte de la ruta de este flujo coincide con una parte del árbol de congestión (en concreto, una cola en el conmutador 8), por lo que aparece *HOL blocking* que limitará el avance del flujo frío. Como puede observarse, paquetes de diferentes flujos (frío y caliente) compartirán una cola "común" en el conmutador 8, y en ella los paquetes de ambos flujos se mezclarán más o menos uniformemente [2]. Aunque estos paquetes tengan destinos distintos, el ancho de banda del enlace de entrada (por el que ambos flujos acceden a la cola común) se repartirá entre los dos flujos. En particular, este ancho de banda se reparte de la siguiente forma: un 33 % del ancho de banda será utilizado por el flujo caliente y un 33 % del ancho de banda será utilizado por el flujo frío. El resto del ancho de banda (33 % del ancho de banda del enlace) se desperdiciará cuando un paquete del flujo caliente esté bloqueado en la cabecera de la cola común en el conmutador 8. Esto conduce a que los paquetes del flujo frío avanzarán hacia su destino a un ritmo menor (33 % de la tasa máxima) en vez de utilizar todo el ancho de banda disponible en este caso (que sería el 66 % del ancho de banda de los enlaces).

En definitiva, el *HOL blocking* introducido por la interacción de los flujos calientes con los fríos es realmente el culpable de que, en situaciones de congestión, disminuya la productividad global de la red y aumente la latencia de los paquetes pertenecientes a

[2]Suponiendo que se emplea, como es habitual, una política equitativa de acceso al enlace.

flujos fríos. Nótese que pueden existir situaciones mucho más complicadas que la reflejada en la figura 3.4, con un mayor número de flujos calientes y fríos. Esto aumentará la intensidad de la contención y el tamaño del árbol de congestión, y por tanto también el número de posibles puntos comunes donde se produzca *HOL blocking*, disminuyendo aún más significativamente la productividad de la red.

El contemplar la congestión desde este punto de vista "alternativo" nos lleva a una observación clave: si fuera posible eliminar el *HOL blocking* que se produce en situaciones de congestión, ésta no afectaría en modo alguno a las prestaciones de la red. Se trataría, pues, no de eliminar la presencia de flujos calientes, sino de impedir su negativa influencia sobre los flujos fríos. La técnica propuesta en la presente tesis parte de esta idea fundamental para resolver el problema de la congestión, y por tanto desde este punto asumiremos que el objetivo principal de nuestra técnica es la eliminación del *HOL blocking*.

La forma en que nuestra propuesta aborda la eliminación del *HOL blocking* será explicada con todo detalle en el siguiente capítulo, pero antes conviene aclarar que existen otras propuestas anteriores orientadas también a la reducción o eliminación del *HOL blocking*. Las ventajas e inconvenientes de estas propuestas se analizan en la siguiente sección.

3.3.2. Técnicas existentes para la reducción o eliminación del *HOL blocking*

El *HOL blocking* es un fenómeno bastante conocido y estudiado desde hace tiempo, por lo que no es de extrañar que se hayan propuesto varias técnicas para solucionarlo. Algunas de ellas tratan de eliminarlo completamente, mientras que otras se limitan a reducirlo en la medida de lo posible. Las más importantes de estas técnicas se describen a continuación.

3.3.2.1. *Virtual Output Queuing*

Como la mayoría de técnicas destinadas a solucionar el *HOL blocking*, las estrategias de la familia *Virtual Output Queuing* (*VOQ*) se basan en dividir la memoria total de cada uno de los puertos de los conmutadores en varias colas independientes, con la intención de almacenar en ellas separadamente paquetes pertenecientes a distintos flujos de datos. La forma exacta en que se distribuye la memoria total entre las colas de cada puerto depende del criterio concreto de almacenamiento de los paquetes que emplee la técnica, lo que determina también, como veremos, si el *HOL blocking* es eliminado totalmente o solamente reducido.

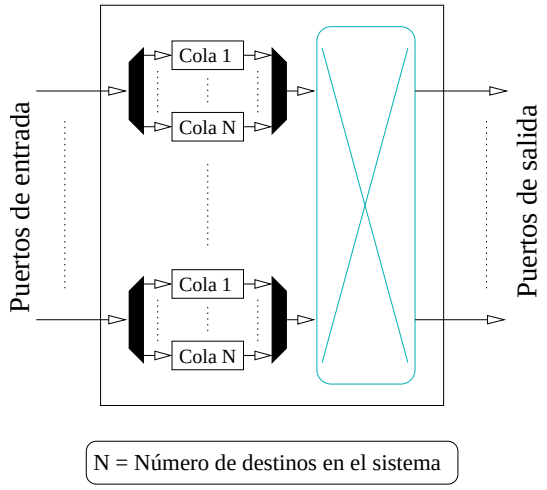


Figura 3.5: Esquema elemental de un conmutador *IQ* que implementa *VOQ* a nivel de red.

Pueden distinguirse dos clases perfectamente diferenciadas de estrategias de tipo *VOQ*: *VOQ* a nivel de red y *VOQ* a nivel de conmutador, que analizamos separadamente a continuación.

***VOQ* a nivel de red.** Cuando se usa *VOQ* a nivel de red [DCD98] (en adelante *VOQnet*), la memoria total de los puertos de los conmutadores se divide en tantas colas como posibles destinos existan en el sistema. Es decir: según este criterio, los conmutadores de una red a la que se conecten N terminales deberían repartir la memoria de cada uno de sus puertos entre N colas distintas. La figura 3.5 muestra un esquema de la organización de las memorias de los puertos en un conmutador que implementa *VOQnet*[3].

Teniendo la memoria de cada puerto la organización mostrada en el esquema, cualquier paquete que llegue a dicho puerto se almacenará en la cola correspondiente a su destino. Esto implica que dos paquetes que tengan destinos diferentes no compartirán nunca la misma cola en ningún punto de la red. Por tanto, si un paquete se bloquea debido a la congestión, no bloqueará a ningún otro paquete que no pertenezca a su mismo flujo. En consecuencia, con una estrategia *VOQnet* no puede existir *HOL blocking* en modo alguno.

La figura 3.6 muestra la misma situación en cuanto a topología y tráfico que la figura 3.4, pero en este caso los conmutadores implementan *VOQnet*. Nótese que existen N destinos en la red ($d1$ a dN), por lo que la memoria de cada puerto se ha dividido en

[3]Para mayor claridad, la figura 3.5, igual que las del resto de la sección, muestra un conmutador *IQ*. Pero téngase en cuenta que la organización de memoria propia de cada técnica expuesta es totalmente aplicable a conmutadores *CIOQ*, con idénticas ventajas e inconvenientes.

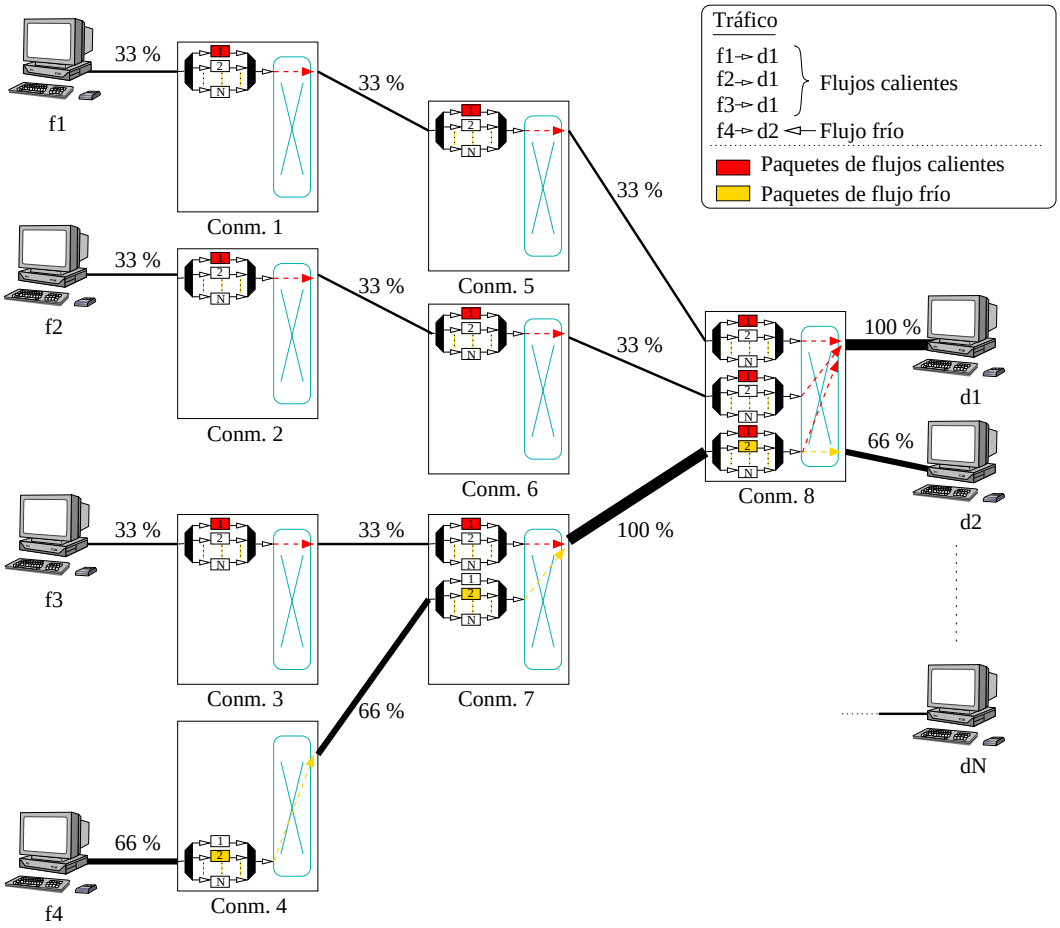


Figura 3.6: Ejemplo de situación con riesgo de *HOL blocking*, resuelto mediante el uso de *VOQnet*.

N colas, y los paquetes de los distintos flujos presentes en la red se almacenan en las colas correspondientes a sus destinos. Como puede apreciarse, en este caso los flujos calientes nunca se mezclan con el flujo frío en ninguna cola, lo que evita que aparezca *HOL blocking*. También puede observarse cómo este hecho repercute positivamente en las prestaciones de la red. Por ejemplo, el enlace entre los conmutadores 7 y 8, por el que se transmiten paquetes pertenecientes tanto a un flujo caliente como a uno frío, se utiliza ahora al 100 % de su capacidad. Y la utilización de los enlaces por donde se envían y reciben los paquetes del flujo frío (enlaces conectados a $f4$ y a $d2$, respectivamente) alcanza el 66 %, lo que supone doblar la utilización conseguida sin ningún mecanismo de control de congestión (ver figura 3.4).

A la vista de este ejemplo, podría pensarse que *VOQnet* es la solución definitiva al problema del *HOL blocking*, ya que lo elimina completamente y permite mantener las prestaciones de la red al máximo incluso en situaciones de congestión. Sin embargo, esta técnica adolece de sus grandes necesidades en cuanto a recursos en la red. En concreto, el número de colas necesarias en los puertos de los conmutadores depende linealmente

del número de terminales conectados a la red. Teniendo en cuenta que aumentar el número de terminales implica también aumentar el número de conmutadores, o de puertos por conmutador, tendremos que el número total de colas necesarias en la red para implementar *VOQnet* crece al menos cuadráticamente en relación con el número de terminales. Esta gran cantidad de colas por puerto plantea dos graves inconvenientes:

- El probable aumento de la memoria total de cada puerto. Aunque se podría pensar en repartir la memoria "original" del puerto entre todas las colas necesarias, existe un tamaño mínimo para cada cola[4]. Si el tamaño de una cola es inferior a este mínimo, el control de flujo podría provocar inanición en la cola. Para que cada cola cumpla con este tamaño mínimo, puede ser necesario aumentar la memoria total del puerto si el número de destinos en la red (y por tanto, el número de colas por puerto) es elevado. Esto se traducirá en un aumento del área de silicio necesaria para implementar la memoria del puerto y, por tanto, en un mayor coste del conmutador.
- La complejidad que supone la gestión de un gran número de colas por puerto. Incluso en el caso de que la memoria total del puerto fuera suficiente para todas las colas, también hay que considerar la dificultad que supone implementar y realizar las funciones de arbitraje, encaminamiento, control de flujo, etc. para un número de colas elevado en cada puerto. Además, las estructuras de datos necesarias para la gestión de cada cola también deben almacenarse en algún tipo de memoria, con lo que, de nuevo, a mayor número de colas, mayor área de silicio se requerirá en cada puerto y mayor será el coste del conmutador.

En definitiva, *VOQnet* es una técnica eficaz pero no escalable. Su implementación real sería costosísima e incluso inabordable a partir de cierto tamaño de la red. Teniendo en cuenta que no es extraño que los sistemas donde se emplean hoy en día las redes de interconexión incluyan miles de terminales, *VOQnet* no puede considerarse una opción viable como técnica de control de congestión.

***VOQ* a nivel de conmutador.** En [AOS93] se propone una alternativa que soluciona la falta de escalabilidad que presenta *VOQnet*, a costa de perder eficacia en cuanto a la eliminación del *HOL blocking*. Se trata de la técnica conocida como *VOQ* a nivel de conmutador (*Switch-level Virtual Output Queuing*, en adelante *VOQsw*).

En este caso, en lugar de definir en los puertos tantas colas como destinos, la memoria de cada puerto se divide en tantas colas como puertos de salida tenga cada

[4]Este tamaño viene determinado por el número de paquetes que pueden salir de la cola durante el intervalo de tiempo que transcurre entre el instante en que el control de flujo informa al emisor de que la cola puede recibir paquetes y el instante en que éstos comienzan a recibirse. Este tiempo, conocido como *round-trip time*, es dos veces el tiempo que tarda la cabecera de un paquete en cruzar el enlace.

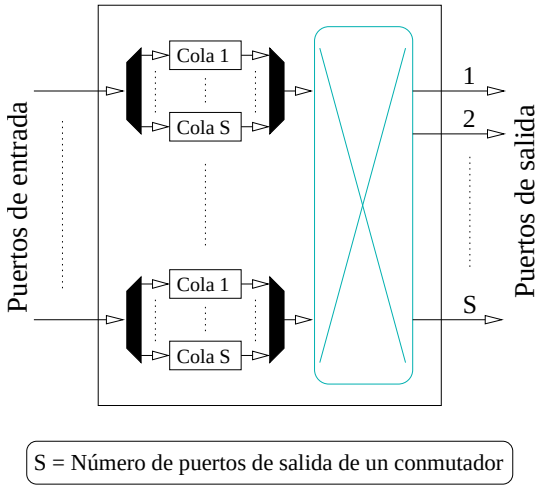


Figura 3.7: Esquema elemental de un conmutador *IQ* que implementa *VOQ* a nivel de conmutador.

conmutador de la red. Es decir: según esta organización de memoria en los puertos, si los conmutadores de la red tienen S salidas, debería repartirse la memoria de cada uno de sus puertos entre S colas distintas. La figura 3.7 muestra un esquema elemental de la organización de las memorias de los puertos en un conmutador que implementa *VOQsw*.

En cuanto al almacenamiento de los paquetes, *VOQsw* establece que, en cada puerto, un paquete debe almacenarse en la cola asignada al puerto de salida solicitado por el paquete. Este criterio de almacenamiento permite resolver algunas situaciones con riesgo de *HOL blocking*. Por ejemplo, la figura 3.8 presenta la misma situación que las figuras 3.4 y 3.6, pero en esta ocasión los conmutadores se ajustan a la técnica *VOQsw*. Puede observarse que todos los conmutadores tienen S salidas, por lo que la memoria de cada puerto se ha dividido en S colas, y cada paquete se almacena en la cola correspondiente a la salida que solicita en cada conmutador. Como puede apreciarse, *VOQsw* también impide en esta ocasión que el flujo frío comparta cola con algún flujo caliente en el conmutador 8, lo que evita que se produzca *HOL blocking*. En este ejemplo, la productividad de la red es idéntica a la conseguida con *VOQnet*, aunque esto se ha conseguido de forma ligeramente distinta en cuanto al almacenamiento de los paquetes en los puertos.

Por otra parte, *VOQsw* es una técnica escalable: la cantidad de colas por puerto no depende del número total de terminales en el sistema, sino del número de puertos de cada conmutador. Esto permite aumentar el tamaño de la red, y la cantidad de conmutadores y terminales, sin que sea necesario aumentar el número de colas por puerto.

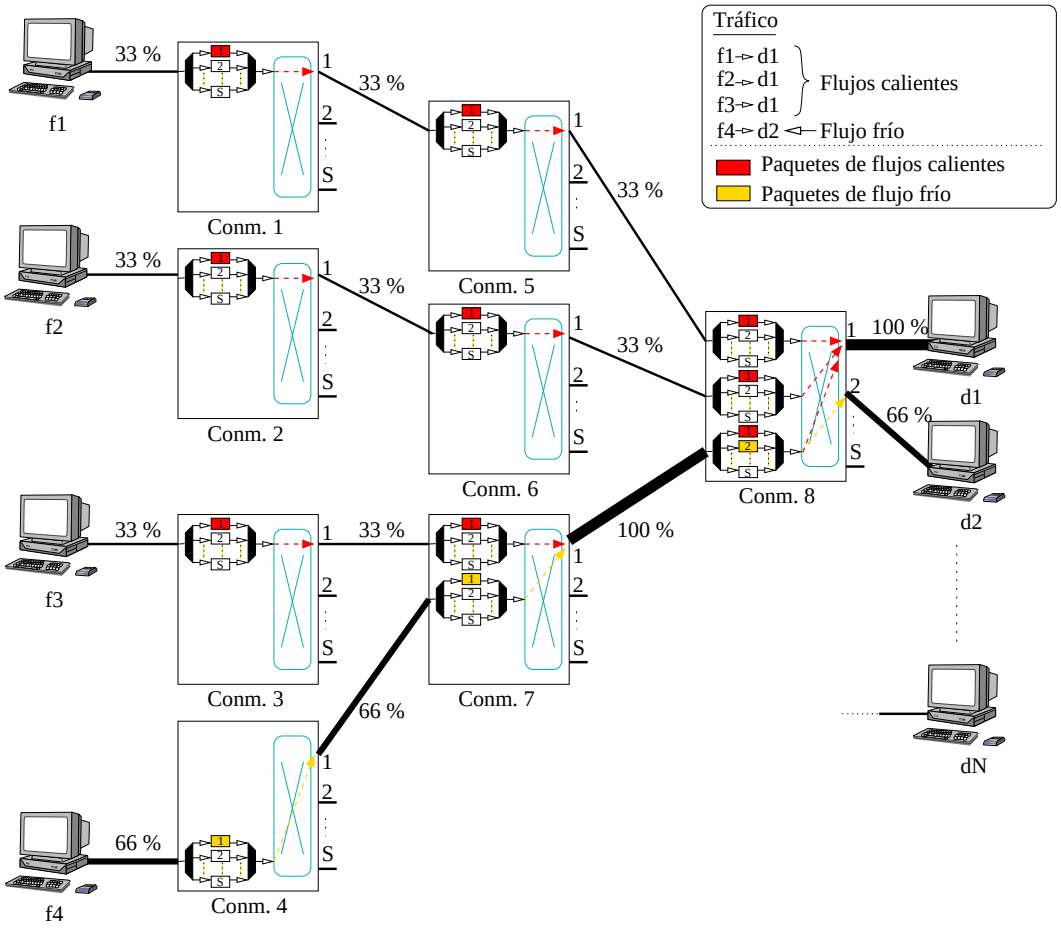


Figura 3.8: Ejemplo de situación con riesgo de *HOL blocking*, resuelto mediante el uso de *VOQsw*.

Sin embargo, debido precisamente a esta filosofía, *VOQsw* sólo es capaz de eliminar con seguridad el *HOL blocking* "local" a un conmutador, entendiendo por tal aquél que se produce por una situación de congestión cuyo origen (la raíz del árbol de congestión) se sitúa en el propio conmutador. Si el árbol de congestión se expande más allá de este conmutador, es posible que exista interacción de flujos fríos y calientes en los conmutadores donde se sitúen las ramas del árbol, y en este caso *VOQsw* no podrá evitar el *HOL blocking* que se produzca. Suele decirse que *VOQsw* elimina el *HOL blocking* de "primer nivel", pero que no es capaz de hacer lo mismo con el *HOL blocking* de "segundo nivel".

Por ejemplo, la figura 3.9 muestra un caso similar, aunque no idéntico, al representado en los ejemplos anteriores. En este caso, la ruta común del flujo frío con uno de los flujos calientes comienza en un puerto del conmutador 7 en lugar de comenzar en el conmutador 8. Puede apreciarse que *VOQsw* es capaz de separar eficazmente el flujo frío del caliente en el conmutador 8, donde se sitúa la raíz del árbol de congestión. Pero en el conmutador 7, ambos flujos compartirán una misma cola al llegar al conmutador

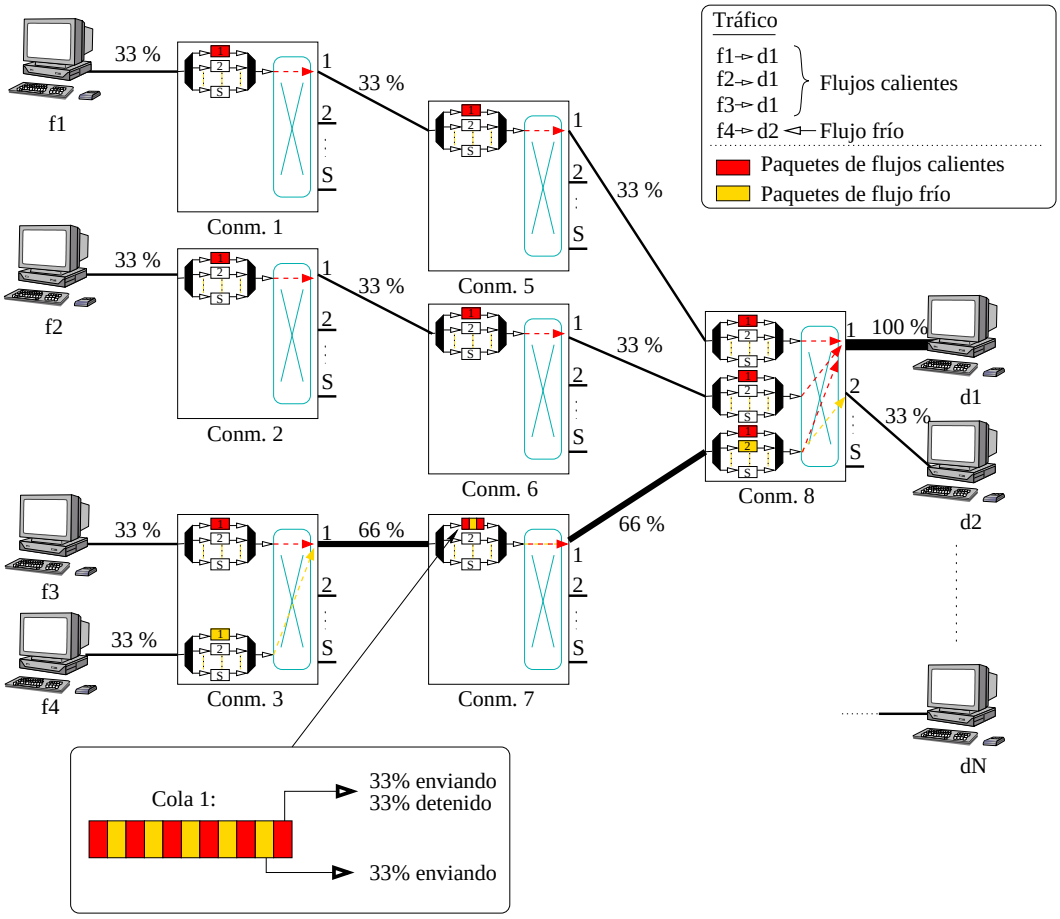


Figura 3.9: Ejemplo de situación con riesgo de *HOL blocking*, que *VOQsw* no puede solucionar.

por el mismo puerto y solicitar el mismo puerto de salida. Esto introduce *HOL blocking* que degrada la productividad de la red a los mismos niveles que los que se obtendrían sin usar ningún tipo de control de congestión. Esto hace inútil, por tanto, la separación de flujos en el conmutador 8. Por otra parte, nótese que *VOQnet* sí resolvería esta situación al separar los flujos también en el conmutador 7, al tener éstos distintos destinos.

En definitiva, aunque *VOQsw* no debería presentar problemas en cuanto a su implementación real, al ser una técnica escalable, no garantiza, como hemos visto, la eliminación del *HOL blocking*, sino que sólo es capaz de manejarlo correctamente en ciertas circunstancias.

Por consiguiente, y aunque se trata de estrategias ampliamente conocidas y difundidas, ninguna de las técnicas de la familia *VOQ* es, a la vez, plenamente eficiente y escalable, y, por tanto, su uso en los sistemas de altas prestaciones basados en redes de interconexión modernas plantearía problemas, en un sentido o en otro.

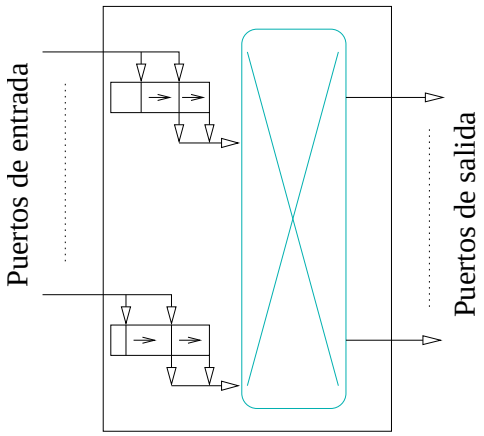


Figura 3.10: Esquema elemental de un conmutador *IQ* con DAMQs.

3.3.2.2. *DAMQs*

En [TF92] se propuso la técnica conocida como "multi-colas dinámicamente asignadas" (*Dynamically-Allocated Multi-Queues, DAMQs*). Se basa en distribuir la memoria total de los puertos en varias colas, pero asignándola a éstas de forma dinámica. Es decir: las celdas de memoria asignadas a cada cola no están determinadas "a priori", sino que se asignan cuando se requieren para almacenar paquetes en dicha cola.

La forma más habitual de implementar las colas es mediante una lista enlazada, que se emplea también para acceder al espacio libre (y por tanto asignable) en la memoria. Todas las colas de un puerto de entrada comparten una única entrada al *crossbar*, que por tanto está multiplexada[5]. La figura 3.10 muestra un esquema elemental de la organización de las memorias de los puertos en un conmutador con *DAMQs*.

El criterio empleado para almacenar paquetes en una cola u otra de las definidas en un puerto determina la utilidad de esta técnica para eliminar el *HOL blocking*. En principio, y utilizando los criterios adecuados, es posible implementar mediante *DAMQs* políticas semejantes a *VOQsw* o *VOQnet*, pero nos encontraríamos con los mismos inconvenientes que con estas técnicas: bien un excesivo número de recursos necesarios, bien una eficiencia limitada.

[5]Existen propuestas de multi-colas dinámicamente asignadas conectadas a *crossbars* de entradas demultiplexadas (*Dynamically-Allocated Fully Connected Queues, DAFCs*) [DB95], pero la implementación real de estos *crossbars* es complicada.

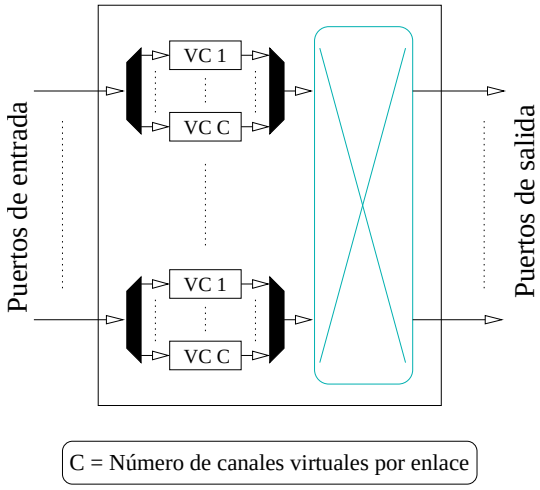


Figura 3.11: Esquema elemental de un conmutador *IQ* con canales virtuales.

3.3.2.3. Canales virtuales

Otra estrategia potencialmente capaz de eliminar el *HOL blocking* es el uso de canales virtuales (*Virtual Channels*) [Dal92]. Se trata de una técnica inicialmente propuesta para redes con conmutación *wormhole*, y consiste en definir varios canales para la transmisión de datos sobre un mismo enlace físico. Como en las técnicas expuestas anteriormente, esta técnica implica la existencia en cada puerto de varias colas separadas. En este caso, existirá una cola para cada canal virtual. La figura 3.11 muestra un esquema elemental de la organización de las memorias de los puertos en un conmutador con varios canales virtuales definidos para cada enlace físico.

El uso de canales virtuales es habitual en entornos donde se diferencia entre distintas clases de tráfico, como pueden ser los sistemas con soporte para calidad de servicio. En este caso, distintas clases de tráfico "circularán" por distintos canales virtuales, lo que en la práctica significa que paquetes de distintas clases se almacenarán en colas distintas. Evidentemente, esto evitará que los flujos de datos asignados a distintos canales virtuales interfieran entre sí. Por tanto, se eliminaría el posible *HOL blocking* entre canales virtuales.

Pero, evidentemente, aquellos flujos asignados a un mismo canal virtual podrán compartir las mismas colas en los conmutadores. Por tanto, para eliminar totalmente el *HOL blocking* sería necesario una organización de los canales virtuales similar a *VOQnet*, con el consiguiente aumento de los recursos necesarios. De otro modo, el uso de canales virtuales sólo puede eliminar parcialmente el *HOL blocking*, de forma similar a *VOQsw*.

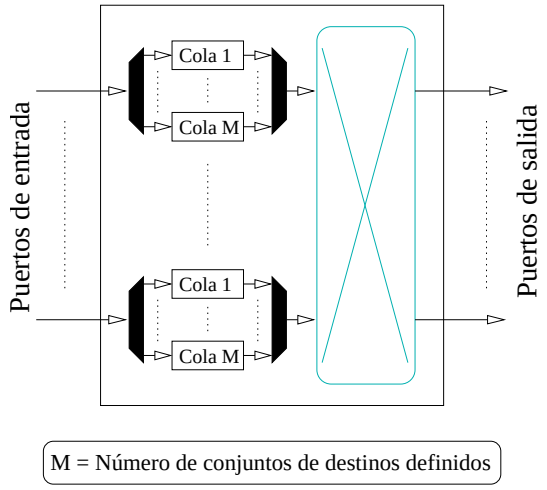


Figura 3.12: Esquema elemental de un conmutador *IQ* que implementa DBBM.

3.3.2.4. *DBBM*

La técnica conocida como *Destination Based Buffer Management (DBBM)* [DFN04] es una propuesta más reciente para reducir el *HOL blocking*. Se basa en dividir el total de los destinos conectados a la red en conjuntos, y asignar en los puertos una cola distinta para cada conjunto. La figura 3.12 muestra un esquema elemental de la organización de las memorias de los puertos en un conmutador que implemente esta técnica.

DBBM define distintos criterios para dividir a los destinos en conjuntos. Por ejemplo, para determinar a qué conjunto pertenece un determinado destino, y suponiendo que se definen M conjuntos, es posible emplear el resultado de aplicar al identificador del destino la operación módulo respecto a M. En cualquier caso, una vez establecido el conjunto correspondiente a cada destino, todos los paquetes con ese destino se almacenarán, en cada puerto de la red, en la cola asignada en ese puerto a dicho conjunto.

Con esta técnica, se elimina el *HOL blocking* que pueda existir entre flujos dirigidos a destinos pertenecientes a distintos conjuntos. Pero es posible que exista *HOL blocking* entre flujos que tengan destinos distintos pero pertenecientes a un mismo conjunto, ya que en ese caso paquetes de estos flujos se podrían almacenar en algún puerto en la misma cola (la correspondiente a dicho conjunto). Sería posible eliminar totalmente el *HOL blocking* si existiera un conjunto para cada destino, en cuyo caso la técnica se comportaría de modo idéntico a *VOQnet*. Pero de nuevo tendríamos que la cantidad de recursos necesaria sería excesiva y por tanto, en la práctica, el número de conjuntos de destinos será limitado, lo que implica que no se podrán resolver todas las posibles situaciones de *HOL blocking*.

Es importante destacar que, en general, se trata de una técnica bastante eficiente, y su comportamiento es mejor que el de *VOQsw*. Pero, a pesar de ello, no puede garantizar la eliminación total del *HOL blocking*.

3.3.2.5. Otras propuestas

Existen otras propuestas de control de congestión destinadas, a diferencia de las técnicas anteriores, a entornos específicos.

Por ejemplo, en [KSS98] se propone un mecanismo de control de congestión para el *Atlas I*, un conmutador para *ATM* implementado en un único *chip*. Este mecanismo se basa en usar un control de flujo basado en créditos (*Credit-Flow-Controlled ATM*) y, sobre todo, en asignar colas separadas para almacenar los paquetes que fluyen hacia destinos congestionados. Esto permite que estos flujos calientes no compartan colas con los flujos fríos, evitando así que se produzca *HOL blocking*. Nótese que, de nuevo, la implementación de esta técnica implica disponer de varias colas en los puertos. Aunque se trata de una técnica eficiente y que consigue una escalabilidad aceptable, presenta el inconveniente de estar diseñada para un entorno muy específico. Además, su escalabilidad no es completa, ya que si los puntos congestionados se producen en el interior de la red, los flujos que crucen dicho punto pueden tener multitud de destinos, y la técnica asignaría colas distintas a cada uno de estos destinos, por lo que el número de colas necesarias puede ser excesivo.

Otra propuesta de control de congestión para un entorno específico puede encontrarse en [KM04]. En este caso, se trata de una técnica diseñada para *Advanced Switching*. Esta estrategia también se basa en mantener varias colas por puerto para separar flujos fríos y calientes. En concreto, se asignan colas separadas para almacenar paquetes destinados a enlaces congestionados, sean o no enlaces conectados a terminales. De nuevo, la escalabilidad es el punto débil de esta técnica, ya que se necesitaría una cola distinta para cada enlace potencialmente congestionado. Además, la técnica está diseñada para gestionar la congestión que se produzca en un conmutador inmediato.

3.3.2.6. Resumen de las características de las técnicas expuestas

A partir de lo expuesto en los puntos anteriores, podemos decir que, en general, todas las técnicas propuestas hasta la fecha para resolver el problema del *HOL blocking* se basan en la existencia de colas independientes en los puertos para almacenar separadamente, o tan separadamente como permita el número de colas, los paquetes pertenecientes a distintos flujos. Esta aproximación parece, en principio, acertada, y de hecho nuestra propuesta se orienta también en este sentido (aunque, como veremos, la técnica que proponemos presenta notables diferencias respecto a las técnicas expuestas

en cuanto a la organización de la memoria y a los criterios de almacenamiento de los paquetes).

Sin embargo, en cuanto a la eficiencia ofrecida por las técnicas expuestas para eliminar el *HOL blocking*, nos encontramos con un dilema: hay que elegir entre que la técnica sea plenamente eficiente y que la técnica sea escalable. Ninguna de las técnicas consigue plena eficiencia sin requerir una cantidad de recursos excesiva.

Por tanto, una técnica de control de congestión que fuera a la vez totalmente escalable y eliminara completamente el *HOL blocking* tendría una utilidad mayor que todas las técnicas expuestas en este capítulo. La principal intención de nuestro trabajo es la obtención de una técnica que cumpla con estos requisitos. Todos los detalles del diseño de esta técnica se explican en los capítulos siguientes.

Capítulo 4

Nueva propuesta para el control de congestión: *RECN*

Habiendo explicado en los capítulos anteriores todos los motivos por los que creemos necesario y oportuno el diseño de una nueva técnica para el control de la congestión, nos centraremos a partir de este punto en describir los detalles concretos de nuestra propuesta.

La técnica propuesta en la presente tesis recibe el nombre de *Regional Explicit Congestion Notification* (*RECN*) y, como se ha indicado ya en capítulos precedentes, con ella se pretende solucionar el problema de la congestión en redes de interconexión de una forma eficiente y escalable. El diseño y prestaciones de *RECN* se analizan en este capítulo y en el siguiente. Concretamente, el presente capítulo se ocupa de todos los aspectos de la técnica básica, y en el próximo se presentan mejoras que la convierten en más eficaz, robusta y sencilla de implementar.

Siguiendo este esquema, este capítulo comienza exponiendo los fundamentos de *RECN*, describiéndose tanto las ideas básicas de la propuesta como el modo en que éstas podrían tomar forma en una implementación real. A continuación se detalla cada aspecto del funcionamiento de la técnica, incluyendo ciertos mecanismos adicionales que, sin estar directamente orientados al control de la congestión, son necesarios para conseguir que *RECN* ofrezca determinadas prestaciones, como la garantía de entrega en orden de los paquetes.

Seguidamente, y para constatar la validez de nuestra propuesta, se presentan resultados que permiten evaluar las prestaciones de *RECN* frente a las de otras técnicas de control de congestión. El método y herramientas empleados en esta evaluación también se describen ampliamente. Por último, el capítulo ofrece una reflexión sobre los resultados obtenidos.

4.1. Planteamiento

A la hora de emprender el diseño de *RECN* se pretendía, como ya hemos explicado, que esta propuesta ofreciera ciertas ventajas sobre otras propuestas para el control de la congestión, en concreto una mayor eficiencia y total escalabilidad. Pero, además, nuestra intención siempre ha sido que *RECN* pudiera implementarse realmente en las actuales tecnologías de redes de interconexión. Debido a esto, las ideas básicas de las que parte el diseño de esta técnica son fruto tanto de la observación de las ventajas e inconvenientes de otras técnicas de control de congestión como de la consideración de las posibilidades de las tecnologías actuales.

En consecuencia, en los siguientes puntos no sólo se detallan las líneas fundamentales en las que se basa nuestra propuesta, sino que también se muestra que nuestro planteamiento es realista, desde el punto de vista de una posible implementación real de la técnica. Además, también se analizan las ventajas potenciales de este planteamiento sobre el de otras técnicas dedicadas a solucionar el problema de la congestión.

4.1.1. Premisas fundamentales

Aunque posteriormente se expondrán con todo detalle cada uno de los aspectos de funcionamiento que pueden distinguirse en *RECN*, es conveniente establecer desde este punto la metodología elemental con la que aborda *RECN* la solución a las situaciones de congestión. Para ello, a continuación se listan las principales ideas que definen la filosofía básica de *RECN* respecto a cómo manejar dichas situaciones:

1. **Solución del problema de la congestión mediante la eliminación del *HOL blocking*.** Como vimos en el capítulo anterior, debido a las características de las actuales redes de interconexión, y a las de los sistemas donde se emplean, no puede considerarse hoy en día el uso de soluciones "drásticas" para la congestión, tales como el descarte de paquetes o el sobredimensionamiento de la red. Por otra parte, las técnicas clásicas específicas para el control de congestión, que se basan en impedir la existencia de la congestión en la red, también plantean problemas en cuanto a su eficiencia y su tiempo de respuesta. Frente a todas estas filosofías *RECN* no considera la existencia de congestión un problema en sí mismo, y no pretende que ésta desaparezca, sino que trata de eliminar su principal efecto negativo: el *HOL blocking*. Por tanto, *RECN* no intenta que desaparezcan de la red los "flujos calientes", que crean puntos congestionados y árboles de congestión, sino que permite que estos flujos existan y procura que no puedan producir *HOL blocking* en los "flujos fríos". De esta manera, aunque exista congestión, las prestaciones de la red no se verán afectadas por ella. Cabe recordar que este enfoque elemental fue analizado ya en el capítulo anterior como

una alternativa a las técnicas proactivas y reactivas, y que es el mismo enfoque adoptado por otras técnicas descritas en dicho capítulo, como *Virtual Output Queuing*, *DBBMs*, etc.

2. **Eliminación del *HOL blocking* mediante el almacenamiento separado de los paquetes pertenecientes a flujos fríos y calientes.** Al igual que otras técnicas destinadas a eliminar o reducir el *HOL blocking*, *RECN* también se basa en intentar que los paquetes pertenecientes a flujos calientes no compartan colas con los paquetes pertenecientes a flujos fríos. En la medida en que esto se consiga, se evitará más o menos el *HOL blocking* que los flujos calientes pueden producir en los fríos. Por otra parte, *RECN* considera que los paquetes pertenecientes a flujos fríos, aun pudiendo seguir rutas parcialmente distintas, pueden almacenarse en una misma cola sin que se produzca entre ellos *HOL blocking* significativo. Nótese que al no ser necesario separar los flujos fríos en distintas colas se consigue un considerable ahorro de recursos. Para separar los flujos calientes de los fríos, *RECN* define en cada puerto un conjunto de colas adicionales específicamente destinadas a contener paquetes cuyas rutas pasen por puntos donde se ha detectado congestión (o sea, paquetes pertenecientes a flujos calientes). Estas colas se denominan *Set Aside Queues* (*SAQs*), y son independientes de las colas estándar, que se utilizan para almacenar paquetes no dirigidos hacia puntos congestionados. De este modo, los flujos calientes no se mezclarían con los fríos en ningún puerto, y por tanto no se darían casos de *HOL blocking*.

3. **Asignación dinámica de las colas destinadas a almacenar paquetes pertenecientes a flujos calientes.** La asignación de las *SAQs* a puntos congestionados se realiza de forma dinámica. Así, en cada puerto existirá un número limitado de *SAQs*, y cada una de ellas puede asignarse a un punto de congestión concreto una vez que el puerto sea informado de la existencia de este punto. Los paquetes que lleguen a un puerto y se dirijan a un punto congestionado para el que haya sido asignada una *SAQ* se almacenarán en dicha *SAQ*, evitando así el posible *HOL blocking* que dichos paquetes podrían provocar en otros flujos. A su vez, si un punto congestionado deja de serlo, las *SAQs* asignadas a dicho punto en distintos puertos pueden liberarse y ser asignadas posteriormente a otros posibles puntos de congestión. Esta política, como se demostrará posteriormente, permite que el número de colas necesarias en cada puerto sea independiente del tamaño de la red, lo que convierte a *RECN* en una técnica totalmente escalable.

4. **Detección de congestión en cualquier punto de la red.** A diferencia de otras técnicas, *RECN* considera que los puntos congestionados pueden situarse en cualquier conmutador de la red y en cualquiera de sus puertos, independientemente de que dicho puerto esté conectado o no a un terminal. Esto permite una separación precisa y eficiente de paquetes en las *SAQs*. Así, por una parte, una misma *SAQ* podrá contener paquetes que, aun pasando todos por el punto

congestionado al que se ha asignado dicha *SAQ*, sigan posteriormente rutas divergentes. Se evita así la asignación de colas distintas para cada destino final, con el consiguiente ahorro de recursos. Por otra parte, una *SAQ* determinada contendrá exclusivamente paquetes que se dirigen con toda certeza hacia un punto congestionado concreto. En definitiva, los recursos destinados a evitar el *HOL blocking* (las *SAQs*) se emplean de forma eficiente.

5. **Propagación de la notificaciones de congestión en paralelo a la formación del árbol de congestión.** Además de poder detectar una situación de congestión en cualquier punto de la red, *RECN* notifica esta situación a todos los puntos de la red por donde crece el árbol de congestión cuya raíz es el punto congestionado detectado. Más aún, esta propagación de la información de congestión se realiza al mismo tiempo que se produce el crecimiento del árbol. Esto implica, por una parte, que a lo largo de las ramas del árbol se asignarán *SAQs* para el punto donde se sitúa la raíz, independientemente de la distancia entre ésta y los puntos notificados. De este modo, los paquetes que forman el árbol de congestión se separarán del resto al almacenarse siempre en *SAQs*, y no producirán *HOL blocking*. Por otra parte, al progresar las notificaciones de congestión al mismo ritmo que el árbol se garantiza una respuesta rápida de *RECN* a las situaciones de congestión.

Evidentemente, este planteamiento general debe reflejarse en mecanismos concretos, cuya implementación no es trivial. En el siguiente punto se analiza si, en principio, esta implementación es posible.

4.1.2. Viabilidad de implementación

Teniendo en cuenta las líneas maestras planteadas, podemos preguntarnos si existen dificultades para implementar en la realidad estas ideas elementales. De lo expuesto en el punto anterior puede deducirse que, a grandes rasgos, la técnica básica debería implementarse por medio de:

- Un mecanismo de detección de congestión.
- Una distribución de la memoria de los puertos en varias colas.
- Una política selectiva de almacenamiento de paquetes en las distintas colas según su ruta.
- Un sistema para asociar colas *SAQs* a puntos congestionados concretos.

Un aspecto que no presenta ningún problema es la detección de la congestión. Ya hemos visto que, por ejemplo, las técnicas reactivas emplean distintos métodos y

criterios para detectar la congestión, por lo que la implementación de un mecanismo similar es perfectamente posible. Así pues, basta, por ejemplo, con monitorizar de algún modo la ocupación de las colas para detectar situaciones de congestión.

Respecto a la estructura de la memoria de datos de los puertos, ya se ha explicado que existen numerosas técnicas que se basan en dividir la memoria de los puertos en varias colas, por lo que realmente nuestra técnica no introduce mayores dificultades en este sentido. En cuanto a la gestión dinámica de las *SAQs*, hay que tener en cuenta que *RECN* tampoco es la primera técnica que plantea algo parecido (recordemos las *DAMQs* [TF92], por ejemplo). De cara a una implementación real, esto puede resolverse, por ejemplo, mediante el uso de una *RAM* de alta velocidad para almacenar los datos y otra *RAM* para almacenar punteros con los que se gestiona cada una de las colas.

En cuanto al almacenamiento de paquetes en una u otra de las distintas colas de un puerto, *RECN* emplea, como hemos explicado, un criterio basado en la ruta seguida por los paquetes. De este modo, una vez que llegue un paquete a un puerto es imprescindible determinar si dicho paquete sigue una ruta que lo llevará a alguno de los puntos congestionados asociados a las *SAQs* activas en dicho puerto, en cuyo caso el paquete se almacenaría en la *SAQ* correspondiente. Caso de no "encajar" en ninguna *SAQ*, el paquete se almacenaría en una cola estándar. Ahora bien, averiguar la ruta que seguirá un paquete es perfectamente posible si se emplea algún tipo de encaminamiento fuente determinista. Si éste es el caso, durante el encaminamiento normal, como sabemos, el conmutador "leerá" parte de la información de encaminamiento contenida en la cabecera del paquete (habitualmente, esta porción de información leída será un desplazamiento entre puertos) para calcular la salida a la que debe encauzarse dicho paquete. Por tanto, podemos asumir que también puede implementarse una lectura de toda la información de encaminamiento contenida en la cabecera del paquete (es decir, puede leerse una secuencia de desplazamientos) para calcular si éste pasará o no por el punto congestionado asociado a una determinada *SAQ*. Cabe recordar que la figura 2.10, en el capítulo 2, representa un ejemplo de encaminamiento en *Advanced Switching* donde se muestra que puede saberse fácilmente con antelación si un paquete cruzará por un punto determinado, utilizando para ello la información de encaminamiento contenida en la cabecera del paquete.

Precisamente la asociación entre *SAQs* y puntos congestionados es uno de los aspectos cuya implementación resulta más interesante. En esencia, se trata de que una parte de la memoria (una cola *SAQ*) se asocie con cierta información (la ruta hasta el punto congestionado). Esta asociación puede implementarse por medio de una memoria direccionable por contenido (*Content-Addressable Memory, CAM*). Una *CAM* permite utilizar una "clave" para acceder a su contenido, en lugar de una dirección como en el caso de las *RAMs*. Por tanto, puede emplearse una *CAM* para complementar cada grupo de *SAQs*. Cada registro de una *CAM* podría contener una ruta

hacia un punto de congestión detectado, y se asociaría cada uno de estos registros a una *SAQ* concreta del puerto. Las rutas almacenadas en los registros serían la "clave" de búsqueda en la *CAM*, lo que permitiría obtener la *SAQ* asignada a un punto de congestión determinado. La forma de expresar una ruta desde el punto donde se sitúa la *CAM* hasta el punto congestionado es sencilla si, de nuevo, asumimos el uso de encaminamiento fuente determinista: basta con utilizar el mismo formato empleado en los paquetes para codificar su ruta en la cabecera. De este modo, puede compararse la ruta de un paquete que llegue al puerto con la ruta contenida en un registro de la *CAM* y si hay coincidencia, almacenar el paquete en la *SAQ* asociada a dicho registro. *RECN* utiliza *CAMs* para almacenar no sólo las rutas hasta puntos congestionados, sino también, como se detallará más adelante, otros datos necesarios para el correcto funcionamiento de la técnica. Por otra parte, la implementación física de *CAMs* es un tema ampliamente estudiado que no supone una dificultad especial.

En conclusión, no parece que en principio existan inconvenientes insalvables para implementar *RECN* tal y como se planteó en el punto precedente. Naturalmente, al haber planteado sólo la "esencia" de la técnica, no nos hemos ocupado todavía de la implementación de otros muchos detalles "menores", pero, como veremos más adelante, tampoco en estos casos se plantean dificultades.

4.1.3. Importancia y ventajas de la propuesta

Aun a falta de explicar con detalle el funcionamiento completo de la técnica, y de mostrar resultados que permitan la evaluación de sus prestaciones, es posible intuir ya las ventajas que, en principio, presenta el planteamiento de *RECN*.

En primer lugar, y a diferencia de las técnicas reactivas clásicas, *RECN* está planteado para que su respuesta a la congestión sea casi inmediata. Esto es debido a que en *RECN* no es imprescindible comunicar a las fuentes de paquetes la existencia de congestión para que se tomen medidas. *RECN* comienza a actuar (asignando *SAQs*) en el mismo conmutador donde se detecta la congestión. Por tanto, *RECN* será más eficiente, y además también evitará que se produzcan las inestabilidades en la red que pueden provocar las técnicas reactivas clásicas.

Por otra parte, y a diferencia de otras técnicas diseñadas para eliminar el *HOL blocking*, *RECN* sí que permite, en principio, garantizar virtualmente que el *HOL blocking* no se produzca, siendo al mismo tiempo una técnica escalable. Recordemos que en el capítulo anterior mostramos diversas técnicas que bien eliminan totalmente el *HOL blocking* a costa de necesitar demasiados recursos, bien requieren recursos limitados a costa de no eliminar totalmente el *HOL blocking*. Las ventajas de *RECN* sobre estas técnicas son debidas, por una parte, a la política de asignación y liberación dinámica de las *SAQs*, que convierte a *RECN* en una técnica escalable (a diferencia, por ejemplo,

de *Virtual Output Queuing* a nivel de red). A su vez, el criterio de asignar *SAQs* a puntos congestionados concretos, situados en cualquier parte de la red, permite eliminar de forma totalmente eficiente el *HOL blocking* que produzcan los "flujos calientes" destinados a dichos puntos (a diferencia, por ejemplo, de *Virtual Output Queuing* a nivel de conmutador).

En definitiva, si se confirma que *RECN* tiene realmente las ventajas que parece presentar a priori, sería una técnica que respondería rápidamente a las situaciones de congestión, eliminando el *HOL blocking* que pudiera producirse, y todo ello independientemente del tamaño de la red. Podría usarse *RECN*, por tanto, en redes no sobredimensionadas, sin riesgo de degradación de prestaciones de la red, aun trabajando cerca del punto de saturación. Y usar menos componentes en la red supondría, como sabemos, un gran ahorro en cuanto al coste de la red y también en cuanto a la potencia consumida por la misma. De ahí la importancia de la técnica planteada.

4.2. Requisitos para su aplicación

Aunque realmente *RECN* podría adaptarse a la mayoría de tecnologías de redes de interconexión, debemos asumir, de cara a proponer un diseño concreto, implementable y detallado de la técnica, que el entorno donde ésta se aplica presenta ciertas características. Estas características, que constituyen los requisitos mínimos para el funcionamiento de *RECN*, se describen en las siguientes secciones.

4.2.1. Información de encaminamiento

Como se ha explicado anteriormente, *RECN* requiere que pueda calcularse en cualquier puerto de un conmutador si un paquete situado en dicho puerto cruzará más adelante por otro punto concreto de la red. Para que esto sea posible, *RECN* asume el empleo de encaminamiento fuente determinista. Así, supondremos que la cabecera de un paquete contiene su ruta, expresada como una serie de desplazamientos entre puertos de los conmutadores, de tal manera que cada conmutador en la ruta emplea uno concreto de estos desplazamientos para encaminar el paquete. También supondremos que un conmutador puede inspeccionar todos los desplazamientos contenidos en la cabecera de dicho paquete, además del que necesita para realizar su encaminamiento. Esto es lo que realmente permitiría conocer la ruta seguida por el paquete a partir de cierto punto.

Aunque no es realmente imprescindible, para mayor facilidad asumiremos que los paquetes emplean el mismo formato de cabecera definido en *Advanced Switching* para el encaminamiento (ver sección 2.7.2). Así, llamaremos *turnpool* a la secuencia de desplazamientos que se incluyen en la cabecera de un paquete y que determinan su

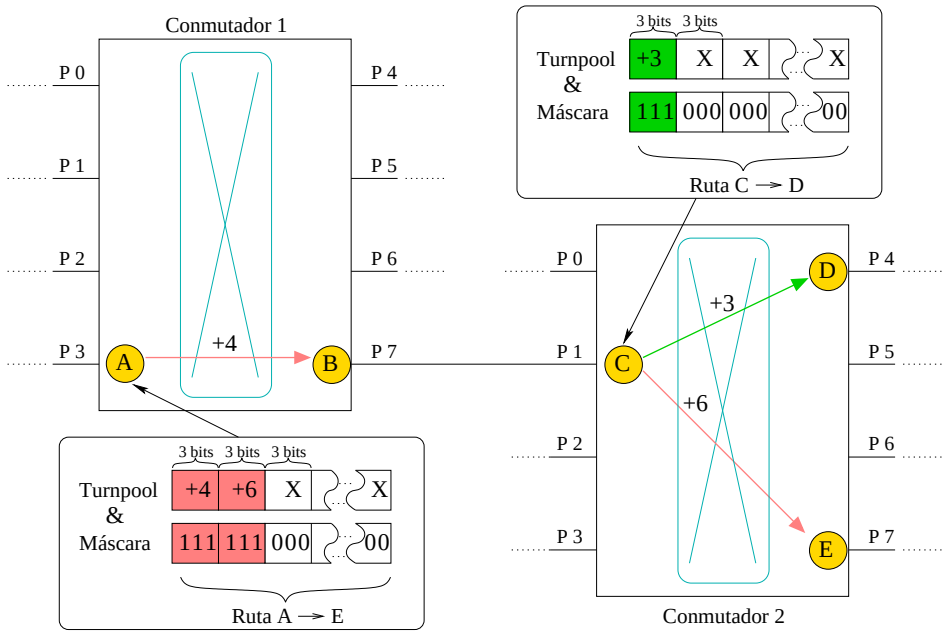


Figura 4.1: Ejemplos del cálculo de la ruta entre dos puntos de la red mediante un *turnpool* y una máscara de bits.

ruta. Recordemos que, a diferencia de otras tecnologías que emplean encaminamiento fuente (como *Myrinet*), los desplazamientos no son eliminados de la cabecera una vez usados por el conmutador correspondiente, sino que existe un *turn pointer* para indicar el siguiente desplazamiento a realizar.

Además, el mismo formato de ruta (un *turnpool*) será empleado para expresar la ruta hasta el punto congestionado que se asocia con cada *SAQ* que se activa tras una notificación de congestión. Así, estas notificaciones contendrán un *turnpool* con la ruta entre el punto notificado y el congestionado. Puesto que estas rutas no tienen por qué ser de la máxima longitud posible, habitualmente no será necesario "rellenar" todos los desplazamientos de este *turnpool*. Más aún, los desplazamientos pueden codificarse mediante distintas cantidades de bits en caso de que existan en la red conmutadores con distinto número de puertos. Por todo ello, es necesario indicar de algún modo qué bits del *turnpool* recibido codifican realmente la ruta que se pretende notificar. Para ello, se asume que las notificaciones también contienen una máscara de bits que permite seleccionar la información significativa del *turnpool*.

En la figura 4.1 se muestran ejemplos de cómo a partir de la información disponible en un punto de la red, en forma de un *turnpool* y una máscara, puede determinarse la ruta entre dicho punto y otro punto concreto. Nótese que la distancia entre ambos puntos y el número de puertos por conmutador determinan la cantidad de desplazamientos "significativos" en el *turnpool* y la cantidad de bits activos (a "1") en la máscara.

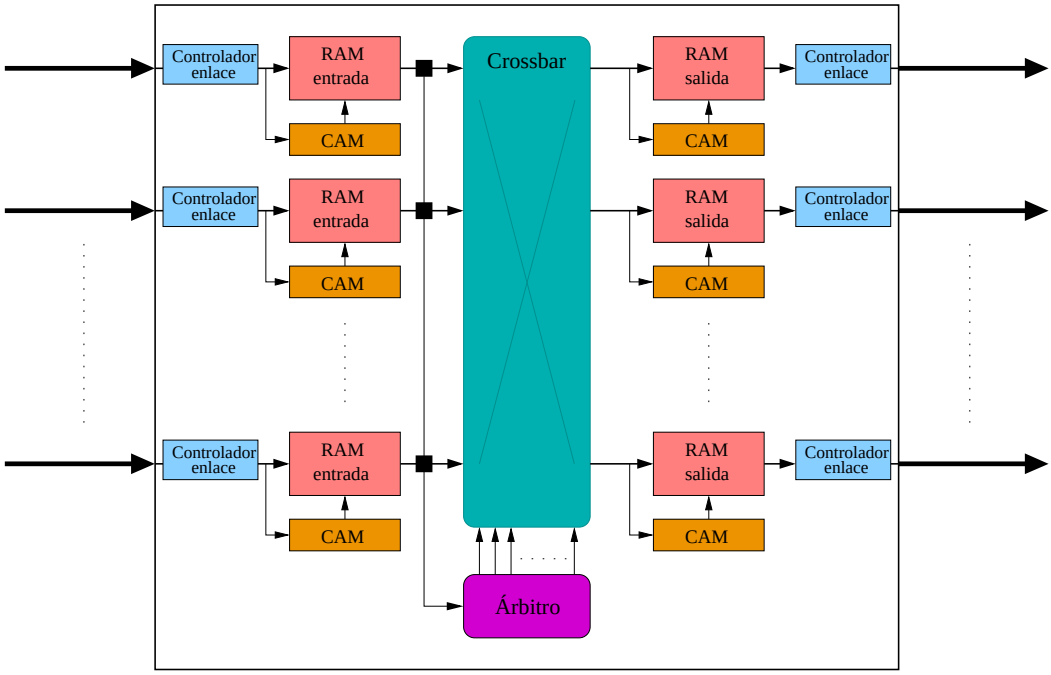


Figura 4.2: Modelo de conmutador asumido por *RECN*.

Téngase en cuenta que la máscara también determina qué cantidad de información del *turnpool* de un paquete debe compararse con la información significativa del *turnpool* asociado a la máscara para saber si el paquete seguirá la ruta indicada por la combinación máscara+*turnpool*. Obviamente, la información del *turnpool* del paquete debe considerarse a partir de la posición indicada por su *turn pointer*.

4.2.2. Arquitectura del conmutador

En lo que respecta a la arquitectura de los conmutadores, se asume el uso de un modelo actualmente habitual en numerosos diseños comerciales. El esquema de dicho modelo puede verse en la figura 4.2. Para mayor claridad, las entradas y salidas del conmutador aparecen separadas en el esquema, aunque podrían estar físicamente juntas si los puertos del conmutador fueran bidireccionales.

Este modelo establece una arquitectura de conmutador basada en un *crossbar* multiplexado tanto a la entrada como a la salida. En cuanto al *speedup* del *crossbar*, asumimos que éste puede variar. De hecho, en las secciones dedicadas a la evaluación de *RECN*, presentaremos resultados para diversos valores del mismo, aunque manteniéndonos siempre en el rango habitual de los conmutadores comerciales: valores de *speedup* entre 1 y 2. Respecto a la técnica de conmutación, se asume el uso de *virtual cut-through*. *RECN* no restringe el número de puertos de los conmutadores, por lo que asumiremos que éstos pueden tener cualquier número de puertos.

En lo referente a las memorias de los puertos, el modelo se ajusta al esquema *CIOQ* por cuanto existen *buffers* en las entradas y salidas del conmutador. En ambos extremos, se asume que las distintas colas que *RECN* define en cada entrada o salida se implementan mediante una única memoria *RAM* para datos y una memoria *RAM* de control para almacenar los punteros necesarios. Recordemos que nuestra propuesta requiere que algunas de estas colas (las *SAQs*) se gestionen mediante ciertos datos almacenados en *CAMs*. Por tanto, el modelo de conmutador también contempla la existencia de una *CAM* en cada entrada y en cada salida del conmutador. De la organización y gestión de todas estas memorias se ocupa con mayor detalle la siguiente sección.

Por último, se asume el uso de una política de arbitraje eficiente, capaz de garantizar un elevado porcentaje de utilización del *crossbar*. En concreto, en nuestras pruebas hemos usado un árbitro basado en gestionar las peticiones de cruce ordenando las peticiones para cada salida según su antigüedad, y dando mayor prioridad a las peticiones de paquetes almacenados en colas estándar frente a las de paquetes almacenados en *SAQs*. Obviamente, la intención de esta última restricción es favorecer el avance de los flujos fríos respecto a los calientes, ya que estos últimos difícilmente conseguirían latencias aceptables, incluso en igualdad de prioridades (dicho de otro modo, se considera a los flujos calientes "casos perdidos" y se los "sacrifica" en beneficio de los fríos). El uso de este árbitro nos permite alcanzar una utilización del *crossbar* de aproximadamente un 95 %. Además, al existir varias colas en cada salida, también debe realizarse un arbitraje adicional que regule el acceso de las distintas colas de una salida al enlace correspondiente. En este caso, la política empleada es *round-robin*, pero también con preferencia de la cola estándar sobre las *SAQs*, por los mismos motivos expuestos anteriormente para el arbitraje del cruce.

4.2.3. Organización de la memoria

RECN emplea distintas colas en cada entrada o salida para separar los paquetes pertenecientes a flujos calientes de los pertenecientes a flujos fríos, por lo que es necesario distribuir la memoria de las entradas y salidas del conmutador en varias colas. Asumiremos que las distintas colas de una entrada o salida se implementan mediante una única *RAM* de alta velocidad destinada a contener los datos de todas la colas, que se controlan mediante punteros almacenados también en memoria *RAM*. En cada entrada o salida, *RECN* establece una única cola estándar, destinada a contener los paquetes pertenecientes a flujos fríos, y un grupo limitado de *SAQs*, donde se almacenarán los paquetes pertenecientes a distintos flujos calientes. La figura 4.3 muestra esta distribución de la memoria *RAM*.

El número de *SAQs* por grupo (n) podría variar en las distintas implementaciones de *RECN*, y de hecho a la hora de su evaluación presentaremos resultados para distintos

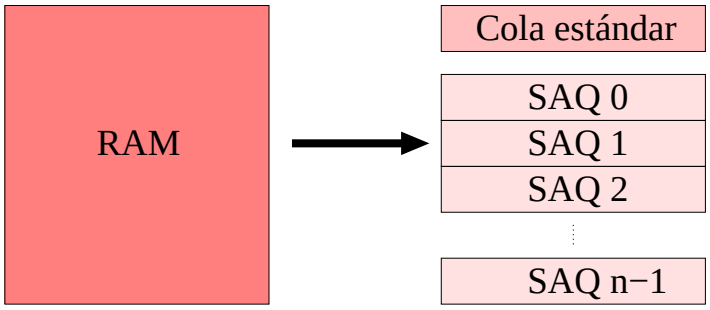


Figura 4.3: Distribución de la *RAM* de una entrada o salida según *RECN*.

valores de n. A título orientativo, el máximo valor de n empleado en nuestras pruebas es 16, aunque el más común es 8. Téngase en cuenta que las *SAQs* se activan sólo tras la llegada de notificaciones de congestión, y que además se liberan cuando la congestión desaparece, por lo que habitualmente el número de *SAQs* activas en la mayoría de entradas y salidas será menor que n. Por otra parte, si se recibe una notificación de congestión en una entrada o salida y ya han sido asignadas las n *SAQs* disponibles, la notificación no tendrá efecto.

Se asume que la memoria *RAM* global de una entrada o salida se reparte de forma dinámica entre las distintas colas definidas en ese punto en un momento dado. Es decir, que ninguna cola tiene un tamaño fijo definido a priori, y las celdas de memoria necesarias para almacenar un paquete se asignan "en caliente" a una cola, una vez determinado que es en dicha cola donde el paquete debe almacenarse. Evidentemente, el control de flujo debe garantizar que no será posible la llegada de nuevos paquetes cuando la suma de los paquetes ya almacenados en todas las colas haya alcanzado la capacidad máxima de la memoria global. Respecto al tamaño de ésta, *RECN* no impone en principio ninguna restricción más allá de las normales para garantizar una operación normal en la red. En nuestras pruebas hemos usado distintos valores para el tamaño de la *RAM* de entradas y salidas, en un rango entre 32 KB y 128 KB.

Además de la memoria *RAM*, *RECN* necesita que exista una memoria *CAM* en cada entrada o salida para almacenar la información utilizada en la gestión del grupo de *SAQs* correspondiente. Cada *CAM* tendrá tantos registros como *SAQs* haya en un grupo, de modo que cada registro estará asociado a una *SAQ* concreta. Estos registros constan de varios campos, destinados cada uno a contener distintos datos sobre la *SAQ* asociada. La figura 4.4 muestra la organización que *RECN* establece para las *CAMs*, incluyendo todos los campos de cada registro.

Como puede apreciarse, uno de estos campos es un bit de habilitación, que indica si la *SAQ* está o no en uso (o lo que es lo mismo, si ya ha sido asignada o no a un punto de congestión). También existen campos para el *turnpool* y la máscara que, como ya se ha explicado, indican la ruta hasta el punto congestionado al que se asigna la *SAQ*.

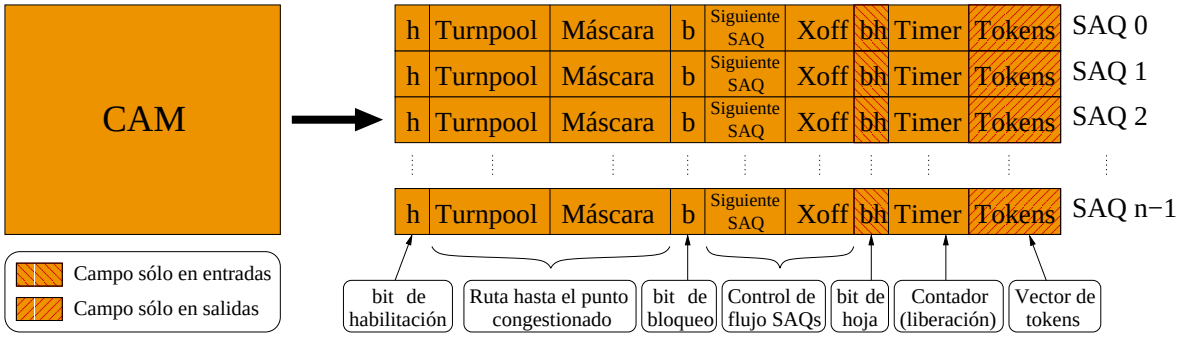


Figura 4.4: Información contenida en los registros de la *CAM*.

Otro bit (bit de bloqueo) indica si existe o no otra cola situada en la misma entrada o salida que bloquea momentáneamente el envío de paquetes desde la *SAQ*; como veremos, este bloqueo es necesario porque permitir a las *SAQs* enviar paquetes justo tras habilitarse puede dar lugar a entregas fuera de orden de los paquetes en destino. Otro par de campos se dedican al control de flujo especial de tipo *Xon/Xoff* que es necesario implementar para las *SAQs*, y que explicaremos con detalle en su momento. Puede adelantarse que uno de estos campos (Siguiente *SAQ*) indica el identificador de una *SAQ* situada en una entrada o salida inmediata en sentido *downstream* respecto a la *SAQ* actual, y que puede enviar a ésta mensajes de control de flujo *Xon/Xoff*. El otro campo dedicado a este control de flujo indica si la *SAQ* se encuentra o no en estado *Xoff* (en caso afirmativo, no podría enviar paquetes). Además, existe un contador (*timer*), que se inicia con cierto valor al habilitarse la *SAQ*, y que se decrementa a cada instante hasta valer cero. Por último, en *CAMs* situadas en las entradas existe un bit que indica si la *SAQ* está en una hoja de un árbol de congestión, mientras que en las *CAMs* de las salidas existe un vector de bits (*tokens*) para controlar a qué entradas se ha notificado congestión desde la *SAQ*. Como se verá, estos datos, junto con el *timer*, indican si se debe liberar o no la *SAQ*.

4.3. Principales aspectos funcionales

Lo explicado en secciones anteriores puede servir para adquirir una idea elemental de cómo opera *RECN* para eliminar el *HOL blocking*, pero realmente el funcionamiento completo del mecanismo es mucho más complejo, y comprende numerosos aspectos que no han sido presentados anteriormente, o lo han sido sólo superficialmente. Con el fin de ofrecer una visión completa y detallada del mecanismo, esta sección muestra, paso a paso, todas las operaciones que deben realizarse en cada una de las fases que pueden distinguirse en el funcionamiento de *RECN*.

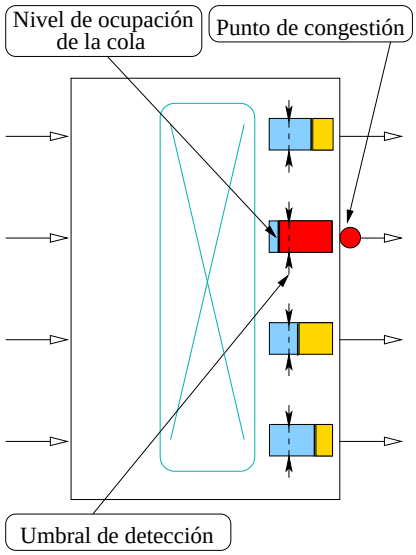


Figura 4.5: Detección de congestión en una salida de un conmutador.

4.3.1. Detección de congestión

Como es obvio, *RECN* se activa debido a la existencia de congestión en algún punto de la red. Por tanto, la detección de dichos puntos puede considerarse el primer paso del mecanismo de cara a eliminar el *HOL blocking* que la congestión pudiera producir.

Para detectar puntos de congestión, *RECN* se basa en monitorizar el nivel de ocupación de las colas estándar situadas en las salidas de los conmutadores. Si dicho nivel supera un umbral, *RECN* asume que existe contención por el uso de dicha salida, y que debido a ello ha aparecido la congestión que se manifiesta en la alta ocupación de la cola estándar. Nótese que este criterio para la detección de congestión asume que los conmutadores poseen un valor de *speedup* mayor que 1 (como, por otra parte, es habitual en conmutadores comerciales), lo cual permite que, en aquellas colas de las salidas por las que exista contención, se acumulen paquetes, al llegar éstos a un ritmo mayor del que pueden enviarse.

La figura 4.5 muestra un ejemplo del sencillo criterio seguido por *RECN* para detectar congestión. Para mayor sencillez, las únicas colas que aparecen en el conmutador de la figura son las colas estándar de las salidas (rectángulos azul claro). En todas estas colas el umbral de detección de congestión se ha indicado mediante una marca. Como puede verse, el nivel de ocupación de la cola estándar en una salida del conmutador (ocupación indicada en rojo en la figura) supera el umbral establecido para detectar congestión, por lo que dicha salida se considera un punto congestionado. Por otra parte, la ocupación de las colas estándar del resto de salidas (en amarillo en la figura) se encuentra por debajo del umbral de detección, por lo que estas salidas no se consideran puntos de congestión. Obviamente, esta situación podría cambiar con el tiempo.

Como es de suponer, el nivel donde debe situarse el umbral de detección influirá en las prestaciones del mecanismo. Un valor demasiado bajo del umbral llevaría a la detección de situaciones de congestión que probablemente no fueran tales. A la inversa, situar el nivel demasiado alto supondría en la mayoría de los casos detectar situaciones de congestión en estado muy avanzado, lo que repercutiría negativamente en la rapidez de respuesta del mecanismo. En definitiva, el valor del umbral de detección debe ajustarse cuidadosamente de cara a obtener un funcionamiento óptimo de *RECN*. De hecho, dicho ajuste, junto con el de otros parámetros críticos, ha constituido el primer paso de las pruebas realizadas para evaluar el mecanismo, como se verá próximamente en la sección correspondiente.

Aunque, como ya se ha indicado, la detección de la congestión se realiza en las salidas de un conmutador, las medidas que *RECN* adopta para remediar la situación no se aplican directamente en la salida congestionada, sino que comienzan a aplicarse en las entradas del conmutador. Esto es evidente si consideramos el propio planteamiento de *RECN*, que trata de separar los flujos congestionados de los flujos fríos. Puesto que un flujo congestionado es, por definición, aquél que pasa por un punto congestionado, es obvio que por dicho punto no pasará ningún flujo frío que pudiera separarse. En cambio, en las colas situadas en las entradas del conmutador pueden existir paquetes dirigidos a la salida congestionada (flujos calientes) o a otras salidas no congestionadas (flujos fríos), por lo que sí que es posible y conveniente en este caso separar ambos tipos de tráfico para eliminar el probable *HOL blocking*. Para que esto pueda realizarse, es necesario, en primer lugar, notificar a las entradas del conmutador la detección de la congestión, lo cual puede considerarse la segunda fase del mecanismo.

4.3.2. Notificación de la detección de congestión

Mientras el nivel de una cola estándar en una salida de un conmutador se sitúe por encima del umbral de detección, cualquier entrada que envíe paquetes hacia esta salida congestionada debe recibir una notificación de detección de congestión[1]. La primera entrada notificada tras una detección es la entrada del conmutador que envió hacia la salida congestionada el paquete cuya recepción provocó que se superara el umbral de detección. En este caso la notificación es inmediata, pero para cualquier otra entrada la notificación se enviará sólo cuando en la salida congestionada se reciba un paquete procedente de dicha entrada.

Esto implica que, una vez que la salida se considera congestionada, debe calcularse el puerto de procedencia de cada paquete que llegue a dicha salida y enviarse una notificación de detección al puerto calculado. Este cálculo es sencillo teniendo en cuenta

[1]En principio, *RECN* establece el contenido de estas notificaciones, pero no restringe el modo en que son enviadas, ni impone la existencia de líneas de control específicas para las mismas, por lo que las notificaciones podrían compartir el ancho de banda con el tráfico convencional.

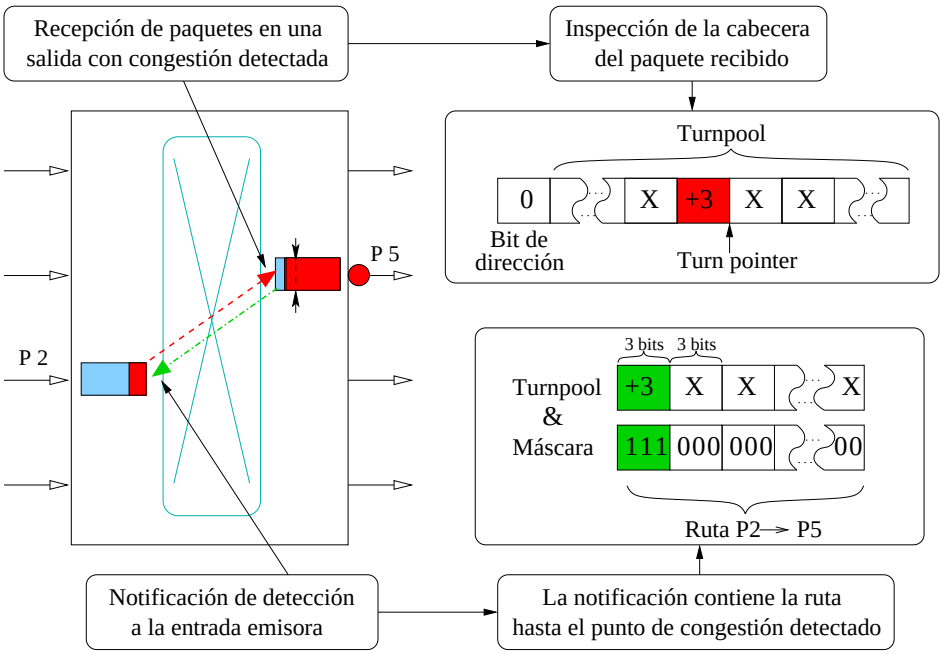


Figura 4.6: Notificación de la detección de congestión a las entradas correspondientes del conmutador.

que el *turnpool* de los paquetes no varía en todo su trayecto, y por tanto basta con buscar en la cabecera de dicho paquete el valor del último salto, que será el valor anterior a la posición actual del *turn pointer*. Una vez obtenido este valor, basta con sumarlo o restarlo (dependiendo del bit de dirección de la cabecera del paquete) al identificador del puerto actual para obtener el puerto emisor del paquete.

Puesto que el objetivo de la notificación es indicar la existencia de un punto congestionado, la notificación contiene la ruta para alcanzar la salida congestionada desde la entrada notificada. Dicha ruta, que en este caso constará de un único salto, se indica, tal y como se explicó en la sección 4.2.1, mediante un *turnpool* y una máscara de bits.

La figura 4.6 presenta un ejemplo de cómo *RECN* reacciona ante la recepción de un paquete en una salida congestionada. Para mayor claridad, sólo se muestran en la figura las colas implicadas. Desde la entrada del puerto $P2$ se envía un paquete hacia el puerto $P5$, cuya cola de salida se encuentra ocupada por encima del nivel de detección. Cuando dicho paquete sea recibido en $P5$, el valor del último salto realizado por el paquete $(+3)$ se restará del identificador del puerto receptor (el bit de dirección está a 0), resultando que el puerto emisor es $P2$. A continuación, se enviará a $P2$ una notificación de detección que incluirá la ruta para alcanzar $P5$ desde $P2$. Como puede verse, al tratarse en este caso de una ruta entre puertos de un mismo conmutador, la máscara de bits indicada en la notificación sólo necesita "seleccionar" los bits correspondientes al primer valor del *turnpool* notificado. A partir de esta recepción de notificación de

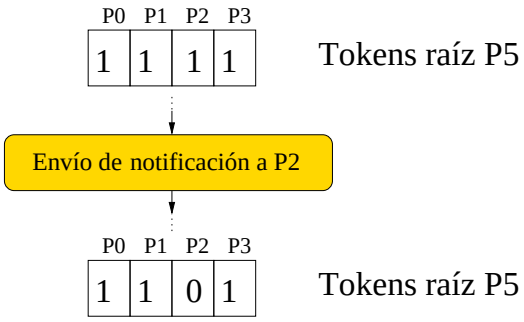


Figura 4.7: Actualización de tokens raíz tras una notificación.

detección de congestión, $P2$ será "consciente" de que la salida de $P5$ es un punto de congestión.

Si en el conmutador existieran varias salidas congestionadas, una entrada cualquiera podría recibir perfectamente notificaciones de detección procedentes de salidas congestionadas distintas, a las que dicha entrada habría enviado paquetes. En este caso, las rutas indicadas en las diferentes notificaciones de detección serían necesariamente distintas, permitiendo a la entrada distinguir entre los distintos puntos de congestión detectados. A la inversa, una salida congestionada enviará notificaciones de detección, como ya se ha indicado, a todas aquellas entradas que le envíen paquetes.

RECN no impide que una entrada pueda enviar sucesivos paquetes a una salida congestionada, incluso tras haber sido notificada dicha entrada sobre la congestión[2]. En este caso, el envío sucesivo de varias notificaciones a la entrada emisora sería redundante, y supondría una sobrecarga en el conmutador. Para evitar el envío de varias notificaciones de detección a una misma entrada desde una salida, *RECN* asocia a cada cola estándar de salida una serie de bits (o *tokens*), tantos como entradas tenga el conmutador. Dichos *tokens* indican, para una salida concreta, qué entradas no han sido aún notificadas desde dicha salida. Tras cada envío de notificación desde una salida, el *token* correspondiente a la entrada notificada se actualiza, impidiendo el envío de más notificaciones hacia dicha entrada desde esa salida. Estos *tokens* reciben el nombre de ***tokens* raíz**, ya que el hecho de que una cola estándar de salida tenga al menos uno de ellos inactivo implica que dicha cola es la raíz de un árbol de congestión [3].

La figura 4.7 muestra la estructura de estos *tokens* en una salida de un conmutador, y la actualización que es necesario realizar para la misma situación reflejada en la figura 4.6 (notificación desde $P5$ a $P2$).

[2]Nótese que esto se ajusta a la idea, indicada repetidamente, de que *RECN* trata de eliminar el *HOL blocking*, pero no la congestión en sí.

[3]No debe confundirse a los *tokens* raíz con la lista de *tokens* incluida en los registros *CAM* asociados a las *SAQs* de las salidas, aun teniendo ambas listas la misma estructura y similares funciones.

Aunque la detección y notificación de la congestión son imprescindibles, no remedian ningún problema por sí mismas. Sólo cuando los puertos de entrada reaccionen ante una notificación, comenzarán a tomarse medidas efectivas contra el *HOL blocking*. Como sabemos, estas medidas consisten en asignar recursos (colas) para separar el tráfico congestionado del no congestionado.

4.3.3. Asignación de recursos

Cuando se recibe una notificación de congestión en una entrada de un conmutador, debe asignarse una *SAQ* de dicha entrada para almacenar los paquetes que, desde ese momento, lleguen a la entrada y se dirijan a la salida congestionada notificada. Una vez asignada a un punto congestionado (o, lo que es lo mismo, a un árbol de congestión cuya raíz se sitúa en este punto), una *SAQ* quedará ligada a dicho punto hasta que se den las condiciones necesarias para que pueda liberarse.

El único requisito necesario para asignar una *SAQ* a un árbol de congestión es que dicha *SAQ* esté "libre", es decir, que no se encuentre ya asignada a otro árbol (recordemos que esto último es perfectamente posible, puesto que un puerto puede ser notificado sobre congestión en distintos puntos). Por otra parte, *RECN* no restringe el orden en que las distintas *SAQs* de un grupo deben asignarse, por lo que cualquier criterio que se emplee para ello debe ser válido (por ejemplo, puede asignarse la primera *SAQ* que se encuentre libre).

Puesto que *RECN* establece que en cada entrada debe existir un grupo limitado de *SAQs*, cabe la posibilidad de que no existan *SAQs* libres en el momento de recibir una notificación. Nótese que, al asignarse una única *SAQ* de cada grupo a un árbol de congestión concreto, las *SAQs* de una entrada sólo se agotarán cuando numerosos árboles de congestión distintos crucen dicha entrada. Si se llega a estos extremos, y no quedan *SAQs* libres en una entrada, las notificaciones que se reciban serán rechazadas y no tendrán efecto alguno. Este rechazo debe comunicarse a la salida que envió la notificación para que pueda volver a intentar notificar la congestión a la entrada, caso de ser necesario. Esto implica que la cola estándar de una salida desde la que se envió una notificación que ha sido rechazada debe reactivar, en su lista de *tokens* raíz, el correspondiente a la entrada notificada sin éxito.

Para llevar a efecto la asignación de una *SAQ*, es necesario que se rellenen ciertos campos del registro *CAM* asociado a dicha *SAQ*. En primer lugar, debe activarse el bit de habilitación de dicho registro, para indicar que la *SAQ* no se encuentra ya libre. Además, deben rellenarse los campos *turnpool* y máscara del registro con el *turnpool* y la máscara incluidos en la notificación de congestión que provocó la asignación de la *SAQ*. Este paso es fundamental, ya que a partir de la información contenida en los campos *turnpool* y máscara se obtendrá la ruta entre la posición de la *SAQ* y su

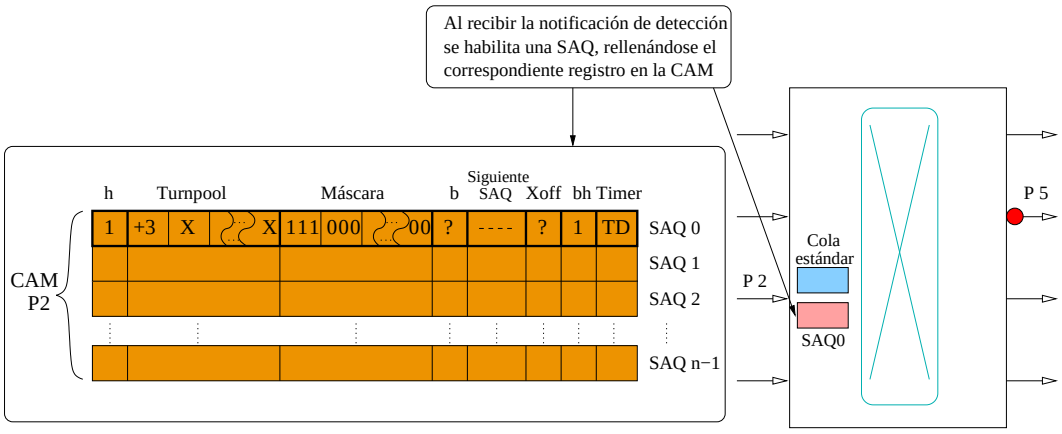


Figura 4.8: Asignación de una SAQ tras una notificación de detección.

punto congestionado asociado, y es por tanto esta información la que realmente liga la SAQ con dicho punto. También debe activarse el bit de hoja, para indicar que en este momento la SAQ se encuentra en una hoja del árbol de congestión cuya raíz se sitúa en el punto congestionado asociado. Por otra parte, debe iniciarse la cuenta atrás del *timer*, desde un valor inicial dado (TD, *Time to Deallocate*). Por último, el bit de bloqueo puede activarse o no, dependiendo de ciertas circunstancias que se explicarán más adelante.

La figura 4.8 muestra un ejemplo de asignación de una SAQ tras una notificación. En concreto, y continuando con el proceso representado en las figuras anteriores, se asume que se acaba de recibir una notificación en $P2$ desde $P5$. Puede comprobarse cómo se rellenan los campos necesarios en el registro CAM asociado a la SAQ asignada ($SAQ0$). Nótese que la información contenida en los campos *turnpool* y máscara indica necesariamente la ruta entre $P2$ y $P5$: un único salto de valor $+3$.

La asignación de alguna SAQ en una entrada de un conmutador implica que en dicha entrada existirán, desde ese momento, varias colas para almacenar los paquetes entrantes: la cola estándar y todas las $SAQs$ activas en ese momento. Ahora bien, $RECN$ trata de eliminar el HOL *blocking* apartando los paquetes dirigidos a puntos congestionados del resto de paquetes, y por tanto una SAQ sólo debe contener paquetes cuya ruta cruce el punto congestionado asociado a dicha SAQ (o, lo que es lo mismo, paquetes pertenecientes al árbol de congestión cuya raíz es dicho punto). Aquellos paquetes que no deban almacenarse en ninguna de las $SAQs$ activas en una entrada (esto es, aquellos paquetes que no pasarán por ninguno de los puntos congestionados asociados a las $SAQs$ activas) se almacenarán en la cola estándar de dicha entrada. En consecuencia, es necesario que se compruebe la ruta de todos los paquetes que lleguen a una entrada donde existan $SAQs$ activas, para determinar si éstos deben almacenarse en alguna SAQ activa o en la cola estándar.

Para calcular si un paquete debe o no almacenarse en una *SAQ* activa, se recurre a una comparación entre la información de encaminamiento contenida en la cabecera del paquete y la contenida en los campos *turnpool* y máscara del registro *CAM* asociado a dicha *SAQ*. Concretamente, la máscara debe aplicarse tanto al *turnpool* del paquete como al *turnpool* del registro *CAM* para seleccionar los bits de ambos *turnpools* que deben compararse. En el caso del *turnpool* del paquete, la máscara debe aplicarse a partir de la posición indicada por el *turn pointer* de la cabecera del paquete, pero en el caso del *turnpool* del registro la máscara se aplica desde el principio del mismo. Una vez seleccionada la secuencia de bits "significativos" para ambos *turnpools*, el paquete se almacenará en la *SAQ* sólo si estas dos secuencias de bits coinciden totalmente. Nótese que la coincidencia indica que el paquete se dirige con toda seguridad hacia el punto congestionado asociado a la *SAQ*.

Téngase en cuenta que la información de encaminamiento de cada paquete que llega a un puerto debe compararse en la forma indicada con el *turnpool* y la máscara asociados a cada una de las *SAQs* activas en la entrada. Si finalmente no hay coincidencia para ninguna *SAQ*, el paquete se almacenará en la cola estándar.

La figura 4.9 muestra un ejemplo de este criterio selectivo de almacenamiento de mensajes en las distintas colas de una entrada con *SAQs* activas. De nuevo, esta figura continúa las situaciones reflejadas en las figuras anteriores, y por tanto se supone que en $P2$ existe una única *SAQ* activa, asignada al punto congestionado $P5$. Pueden observarse las acciones que tienen lugar tras la llegada de distintos paquetes a $P2$. Al comparar la ruta seguida por el paquete A con la ruta asociada a la *SAQ0* se encuentra una coincidencia, y por tanto, el paquete A se almacenará en dicha *SAQ*. En cambio, al comparar la ruta seguida por el paquete B con las rutas asociadas a las *SAQs* activas (en este caso, sólo *SAQ0*) no se encuentra coincidencia alguna, y por tanto el paquete B se almacenará en la cola estándar.

Esta política de asignar *SAQs* para el almacenamiento separado de los paquetes dirigidos a puntos congestionados es la que permite a *RECN* garantizar que dichos paquetes no producirán *HOL blocking* sobre el resto de paquetes. Por ejemplo, en la figura 4.9, los paquetes que se dirijan al punto congestionado $P5$ no pueden producir *HOL blocking* sobre ningún paquete almacenado en la cola estándar de $P2$ al estar almacenados aparte, en la *SAQ0*.

Por otra parte, nótese que la cola estándar puede almacenar paquetes que sigan rutas total o parcialmente distintas. Ahora bien, el hecho de que se encuentren en dicha cola implica que ninguno de ellos se dirige a un punto congestionado (en caso contrario estarían almacenados en alguna *SAQ* activa en dicho punto) y, por tanto, según las premisas seguidas por *RECN*, ninguno de ellos debería introducir *HOL blocking* de forma significativa.

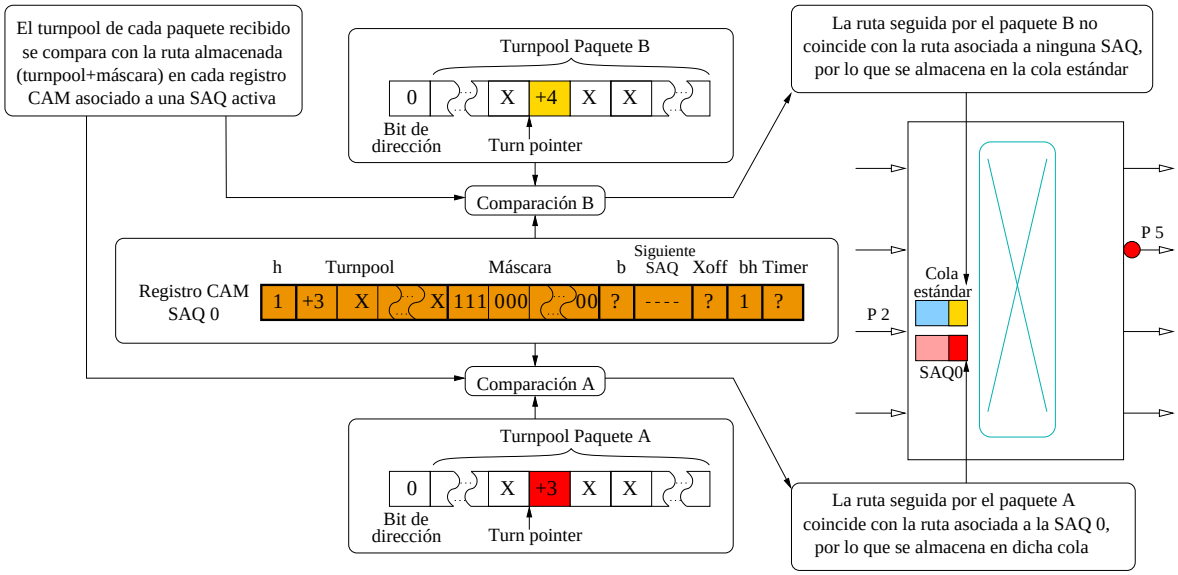


Figura 4.9: Almacenamiento de paquetes en distintas colas de entrada en función de su ruta.

En definitiva, mediante el almacenamiento selectivo y separado de paquetes en colas estándar y *SAQs* se elimina el *HOL blocking* que pueda producirse. Sin embargo, hasta ahora sólo hemos considerado cómo *RECN* asigna recursos en las entradas de un conmutador para eliminar el *HOL blocking* que pudiera producir la congestión de una salida en el mismo conmutador. Pero, como vimos en la sección 2.6.2, la congestión tiende a expandirse formando árboles de congestión que cruzan varios conmutadores de la red. En estos casos, los paquetes pertenecientes a un árbol de congestión podrían seguir produciendo *HOL blocking* más allá del conmutador donde se sitúa la raíz del árbol. Por tanto, si se desea eliminar totalmente el *HOL blocking*, deben asignarse recursos para almacenar por separado paquetes congestionados en todos los puntos de la red por donde circulen dichos paquetes. Así pues, en general, no basta con detectar congestión en una salida de un conmutador y comunicarlo a sus entradas: es necesario comunicar también la existencia del punto de congestión a todos puertos de otros conmutadores por donde se expanda el árbol cuya raíz es dicho punto, para que en estos puertos, a su vez, se tomen las medidas oportunas.

4.3.4. Propagación de la información de congestión

Con el objetivo de que la posición de un punto de congestión detectado en un conmutador pueda ser comunicada a puertos situados en otros conmutadores, *RECN* establece que cualquier *SAQ*, si es necesario, debe notificar congestión, indicando la ruta hasta su punto de congestión asociado, a cualquier puerto del que reciba directamente paquetes. Téngase en cuenta que estas notificaciones enviadas desde *SAQs* no suponen

la detección de nuevos puntos de congestión, sino la difusión de la posición de puntos de congestión detectados anteriormente.

Para determinar si una *SAQ* debe notificar congestión, *RECN* emplea, al igual que en la detección de nuevos puntos de congestión, un criterio basado en monitorizar el nivel de ocupación del *buffer* correspondiente. Así, si el nivel de ocupación de una *SAQ* de un puerto supera cierto nivel, es porque un puerto situado inmediatamente en sentido *upstream* está enviando una gran cantidad de paquetes que, al llegar al puerto donde se sitúa la *SAQ*, se almacenan en ésta. Esto implica necesariamente que el puerto *upstream* inmediato está siendo cruzado por numerosos paquetes que pertenecen al árbol de congestión cuya raíz es el punto de congestión asociado a la *SAQ*. Y puesto que es evidente que el árbol de congestión está presente en el puerto inmediato, éste debería conocer la posición de la raíz de dicho árbol de cara a poder separar los paquetes realmente congestionados. Para ello, en definitiva, *RECN* establece que una *SAQ* cuyo nivel de ocupación supere cierto umbral (umbral de propagación) debe notificar congestión a aquel puerto *upstream* inmediato que le envíe paquetes, incluyendo en dicha notificación la ruta (como siempre, en forma de un *turnpool* y una máscara de bits) hasta la raíz del árbol de congestión asociado.

La recepción de una notificación de congestión procedente de una *SAQ* debe producir, a su vez, la asignación de una nueva *SAQ* en el puerto notificado, de forma similar a lo que sucede tras recibirse una notificación de detección de congestión desde una cola estándar. Y como en ese caso, las operaciones necesarias para asignar la nueva *SAQ* consisten en rellenar campos del registro *CAM* asociado a dicha *SAQ*. Así, el bit de habilitación debe activarse, y los campos *turnpool* y máscara deben rellenarse con la información correspondiente a la ruta indicada en la notificación recibida. Por supuesto, la información contenida en estos dos últimos campos se empleará para determinar si los paquetes que llegan al puerto deben almacenarse o no en la nueva *SAQ*. También el *timer* correspondiente debe iniciarse. Por otra parte, al tratarse de una *SAQ* asignada en una salida, no existe un único bit de hoja en su registro *CAM*, y en su lugar deben activarse todos los bits de la correspondiente lista de *tokens*. Además, en este caso, debe rellenarse en el registro *CAM* el campo "Siguiente *SAQ*" con el identificador de la *SAQ* que notificó congestión. Para ello, este identificador también debe incluirse en la notificación. De este modo, si una *SAQ* se ha asignado por una notificación procedente de otra *SAQ*, puede saberse que esta última será concretamente la cola que reciba directamente los paquetes que la primera envíe. Como veremos, esto es de suma importancia para el control de flujo especial que establece *RECN* entre *SAQs*.

Nótese que el envío de una notificación de congestión desde una *SAQ* en la entrada de un conmutador supondrá la asignación de una *SAQ* en una salida de otro conmutador, a diferencia de una salida en la que se detecta un nuevo punto de congestión, donde no se asignan *SAQs* directamente. Téngase en cuenta, además, que tras una detección de un nuevo punto de congestión se envía una notificación interna, desde la salida

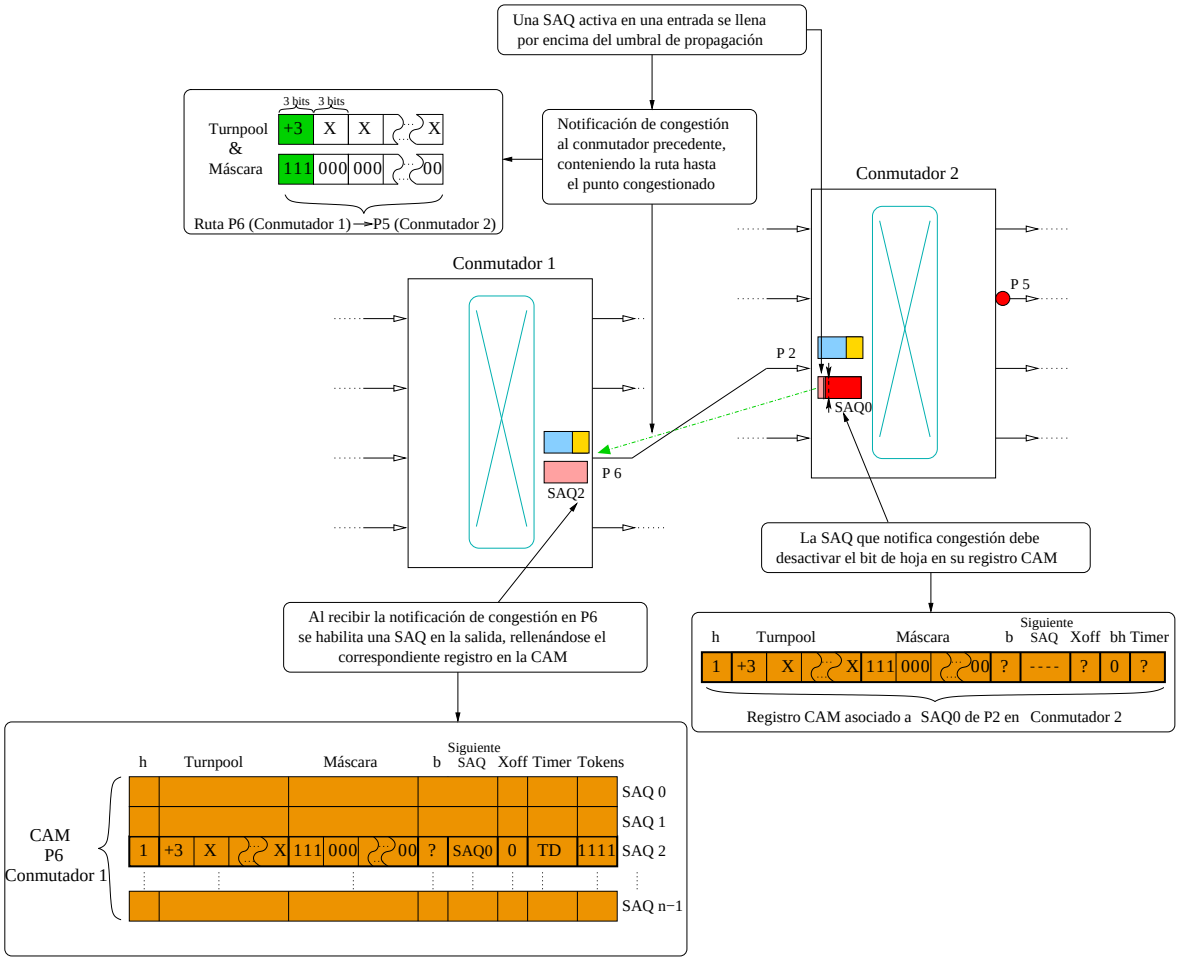


Figura 4.10: Asignación de una *SAQ* en una salida tras una notificación de congestión procedente de una *SAQ* en una entrada de otro conmutador.

congestionada a una entrada en el mismo conmutador, pero para notificar congestión desde una *SAQ* en una entrada, es necesario el envío de un paquete de control que cruzará el enlace entre los dos conmutadores implicados. Cabe destacar que una salida sólo puede recibir notificaciones desde una única entrada, y por tanto no puede haber más *SAQs* activas en una salida que en la entrada a la que se conecta. Esto implica que las notificaciones de congestión que se reciban en una salida nunca serán rechazadas, ya que siempre habrá suficientes *SAQs* para asignar alguna tras una notificación.

Una *SAQ* situada en una entrada debe desactivar el bit de hoja de su registro *CAM* asociado cuando notifique congestión, puesto que en ese caso dicha *SAQ* ya no se encuentra en ninguna hoja del árbol de congestión asociado. Para evitar el envío de notificaciones de congestión redundantes, una *SAQ* de una entrada con el bit de hoja desactivado no enviará notificaciones de congestión, independientemente de su nivel de ocupación.

La figura 4.10 muestra un ejemplo de notificación de congestión desde una *SAQ* en la entrada de un conmutador (conmutador 2) a una salida de un conmutador distinto (conmutador 1). El punto de partida de la figura es de nuevo la situación reflejada en figuras anteriores: la raíz del árbol detectado se sitúa en $P5$ en el conmutador 2, y ya existe una *SAQ* (*SAQ0*) asignada a este punto en $P2$ en el mismo conmutador. Esta *SAQ*, según la figura, se ha llenado hasta alcanzar el nivel de propagación, y por tanto debe enviar una notificación de congestión que será recibida en $P6$ en el conmutador 1. La notificación contiene la ruta hasta $P5$, conocida por la *SAQ0* de $P2$ gracias a la información contenida en los campos correspondientes de su registro *CAM*. Como puede observarse en la figura, la recepción de la notificación en el conmutador 1 provoca la asignación de una *SAQ* en $P6$ (SAQ2), rellenándose el registro *CAM* correspondiente. Puede observarse que en este registro el campo "siguiente *SAQ*" indica que la *SAQ* que originó la asignación es la *SAQ0* de la entrada situada en sentido *downstream*. También es de destacar que la nueva *SAQ* asignada en $P6$ tiene activos todos los bits de la lista de *tokens* de su registro *CAM*, mientras la *SAQ* notificante desactiva su correspondiente bit de hoja. Por último, obsérvese que la información de los campos *turnpool* y máscara es la misma en los registros *CAM* de la *SAQ* que notifica y de la *SAQ* recién asignada. Esto es debido a que un paquete sólo realiza un salto entre una entrada y una salida de un mismo conmutador, pero no entre una salida de un conmutador y la entrada del siguiente; por tanto, la ruta hasta una salida en un conmutador, expresada en saltos, es la misma desde una entrada en el conmutador y desde la salida del conmutador precedente que está conectada a dicha entrada.

Por otra parte, las *SAQs* situadas en las salidas cumplen plenamente la norma sobre notificaciones expuesta en los primeros párrafos de esta sección y, por tanto, si su nivel de ocupación supera el umbral de propagación, deben notificar congestión a todas aquellas entradas que les envíen paquetes. Estas notificaciones son, lógicamente, internas (se producen entre salidas y entradas de un mismo conmutador) y, además, individuales: cuando se recibe un paquete en una *SAQ* de salida y el nivel de ocupación se encuentra por encima del umbral, sólo se envía notificación a la entrada de origen del paquete, no a todas las entradas del conmutador. El motivo de esta política, lógicamente, es notificar congestión sólo a aquellas entradas por donde circule el árbol de congestión. Obviamente, la entrada emisora de un paquete puede determinarse fácilmente en la salida receptora comprobando en el *turnpool* del paquete el último salto realizado.

Siguiendo la pauta del mecanismo, la recepción de una notificación de congestión en una entrada desde una *SAQ* de salida provocará que se asigne en la entrada notificada una nueva *SAQ* mediante el procedimiento habitual de rellenar el registro *CAM* correspondiente. Nótese que, en este caso, sí puede rechazarse una notificación por falta de *SAQs* libres en la entrada, en cuyo caso no se asignará ninguna *SAQ* en la misma. Además de esta última, existen otras dos importantes diferencias respecto a las notificaciones de congestión desde una *SAQ* de entrada hasta una salida.

La primera diferencia se debe a que, mientras que una *SAQ* a la entrada de un conmutador sólo puede recibir paquetes de un puerto, una *SAQ* en una salida puede recibir paquetes de múltiples puertos, y por tanto puede enviar notificaciones de congestión a varios puertos. En consecuencia, si se quiere evitar el envío de notificaciones duplicadas a la misma entrada desde una *SAQ* de salida, es necesario controlar de algún modo que entradas han sido ya notificadas. Para ello, como ya se ha mencionado, cada *SAQ* en una salida cuenta con un campo en su registro *CAM* correspondiente, conteniendo una lista de *tokens* (bits) cuya funcionalidad es similar a la de la lista de *tokens* raíz asociada a las colas estándar. Es decir, una *SAQ* de una salida tendrá tantos *tokens* como entradas tenga el conmutador, y cada vez que envíe una notificación se desactivará el *token* correspondiente a la entrada notificada. Así, una *SAQ* de salida enviará notificaciones sólo a aquellas entradas con el correspondiente *token* activo. Evidentemente, si una entrada rechaza una notificación por no disponer de *SAQs* libres, el *token* correspondiente deberá reactivarse. Nótese que, con este criterio, una *SAQ* de salida puede considerarse situada en la hoja de un árbol de congestión sólo si todos sus *tokens* están activos. Por el contrario, una *SAQ* asignada en una entrada tras recibirse una notificación de congestión pasa a estar en una hoja del árbol, y para indicar esta condición es suficiente con tener activo un único bit (el correspondiente bit de hoja).

La segunda diferencia afecta a la información de ruta que se envía en la notificación. En ella debe indicarse la posición de la raíz del árbol respecto de la entrada notificada, pero ésta se encuentra "un salto más lejos" de la raíz que la salida donde está la *SAQ* que notifica. Por tanto, para componer la ruta que se incluye en la notificación, debe añadirse el valor de este salto al *turnpool* almacenado en el registro *CAM* de la *SAQ* notificante y, además, actualizar la máscara de bits para que también se consideren significativos los bits del salto añadido.

La figura 4.11 muestra un ejemplo de notificación de congestión desde una *SAQ* en la salida de un conmutador (*SAQ2* en $P6$, conmutador 1) a una entrada ($P1$) en el mismo conmutador. Como en figuras anteriores, la raíz del árbol de congestión para el que se asignan todas las *SAQs* representadas se encuentra en el conmutador 2 (en $P5$, concretamente). Puede apreciarse que la *SAQ2* de $P6$ en el conmutador 1 se encuentra ocupada por encima del umbral de propagación, por lo que, al recibir paquetes desde $P1$, debe notificar congestión a dicho puerto. La figura también muestra la ruta incluida en la notificación, que, como puede observarse, contiene un salto significativo más (el salto entre $P1$ y $P6$, de valor $+5$) que la ruta almacenada en el registro *CAM* asociado a la *SAQ* notificante. Esta ruta se almacena en los campos correspondientes del registro *CAM* asociado a la nueva *SAQ* que se asigna en $P1$ tras la recepción de la notificación (*SAQ1*). Por último, cabe destacar cómo la lista de *tokens* del registro *CAM* asociado a la *SAQ2* de $P6$ en el conmutador 1 se actualiza tras el envío de la notificación, desactivándose el *token* correspondiente a la entrada notificada ($P1$, en este caso).

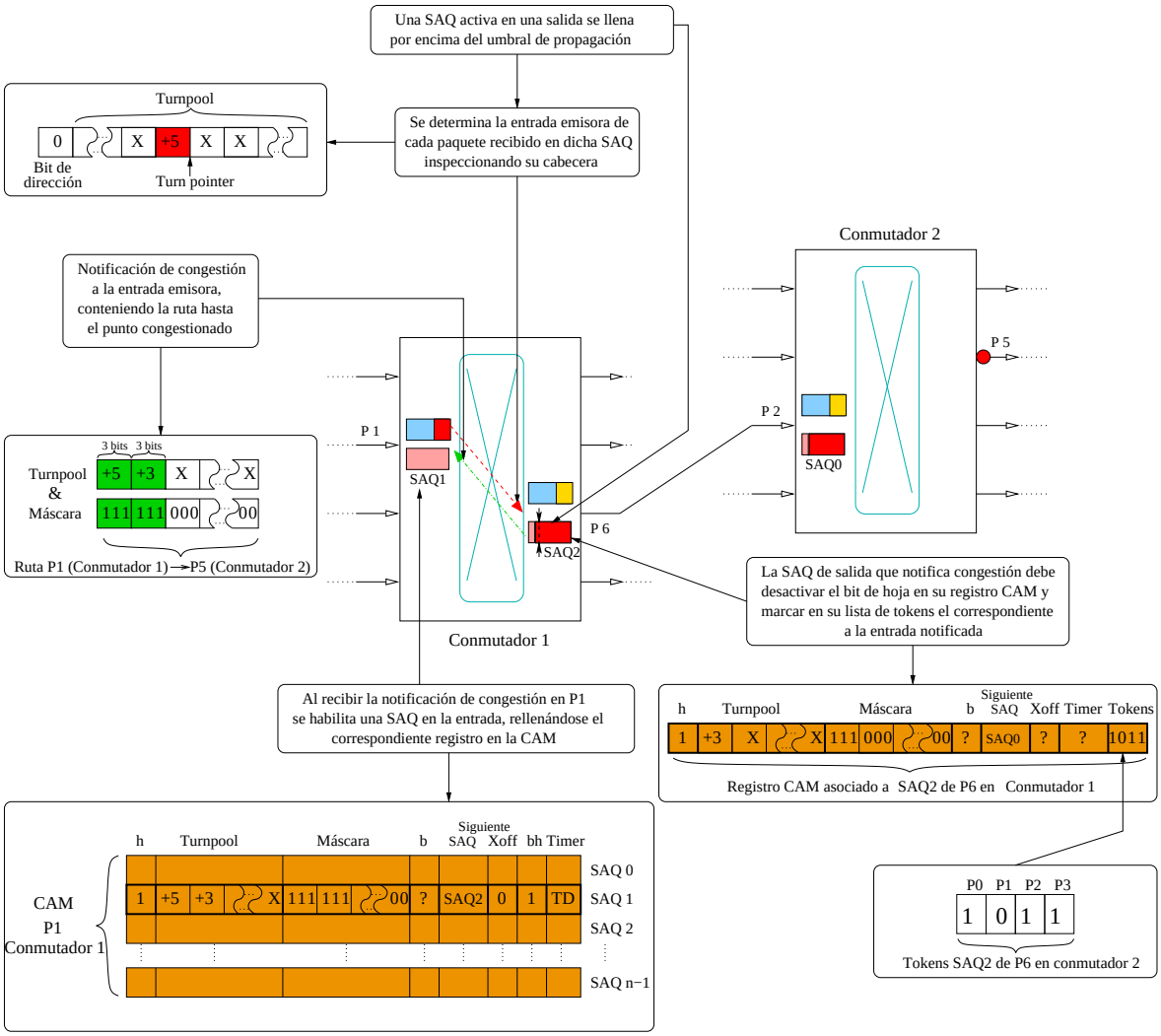


Figura 4.11: Asignación de una *SAQ* en una entrada tras una notificación de congestión procedente de una *SAQ* en una salida del mismo conmutador.

Por supuesto, las *SAQs* asignadas en las entradas a consecuencia de una notificación de congestión enviada desde una *SAQ* de una salida podrían seguir mandando notificaciones de congestión en sentido *upstream*, y así sucesivamente. Nótese que, al alejarse las *SAQs* que notifican de la raíz, la información sobre la ruta incluida en las notificaciones constará de un número creciente de saltos significativos. De este modo, la información que permite ubicar la posición de la raíz de un árbol de congestión se va actualizando y propagando en sentido *upstream* por todas las ramas de dicho árbol. Al final de este proceso, cualquier puerto de la red por donde circule el árbol será consciente de esta circunstancia, y además conocerá la ruta hasta la raíz del árbol, independientemente de la distancia entre dicho puerto y la raíz. En consecuencia, en todos los puertos situados en las ramas de un árbol de congestión se podrán asignar *SAQs* para contener exclusivamente los paquetes dirigidos a la raíz de dicho árbol. Se

elimina así totalmente el *HOL blocking* que los paquetes pertenecientes al árbol, estén donde estén, pudieran producir.

Con la intención de ofrecer una visión global de la actuación de *RECN*, la figura 4.12 representa, a diferencia de las figuras anteriores, una red completa (en concreto una red multietapa) donde se ha formado un árbol de congestión, lo que ha provocado a su vez la correspondiente reacción de *RECN*. La figura refleja el resultado final de las operaciones del mecanismo, en un momento en que ya se han asignado *SAQs* a lo largo de todas las ramas del árbol para almacenar los paquetes que lo forman. De cara a facilitar el seguimiento del proceso de asignaciones de *SAQs*, la figura muestra también todas las notificaciones que han dado lugar a estas asignaciones, aunque debe tenerse en cuenta que las notificaciones (y asignaciones) no se producen simultáneamente, sino en sucesivas fases. Puede comprobarse cómo las notificaciones se propagan por la red desde la raíz del árbol hasta las hojas del mismo, ajustándose fielmente a todas las ramas.

Como puede observarse en la figura 4.12, no existe ninguna cola en la red donde se almacenen a la vez paquetes congestionados y paquetes no congestionados (o flujos fríos). Por ejemplo, aunque los paquetes del flujo frío 2 coinciden con los paquetes congestionados en la entrada y la salida del conmutador 3 y en la entrada del conmutador 5, no siguen una ruta que lleve al punto congestionado, y por tanto se almacenan en colas estándar en todo su trayecto, totalmente separados de los paquetes congestionados, almacenados siempre en *SAQs*. Lo mismo sucede con los paquetes del flujo frío 1, que circulan "en paralelo" a una rama del árbol de congestión durante casi todo su trayecto, el cual sólo se aparta del árbol en el último salto. Nótese que, si no existieran las *SAQs*, los paquetes de estos flujos fríos se almacenarían en la misma cola que los paquetes congestionados en aquellos puntos de la red donde sus trayectos coinciden, y en estos puntos se produciría *HOL blocking*. Por tanto, se comprueba cómo *RECN* elimina totalmente el *HOL blocking* que pudieran producir los paquetes del árbol, en todos los puntos de la red por donde éstos circulan.

Por último, conviene aclarar que, aunque en la anterior figura la raíz del árbol de congestión se sitúa en una salida de un conmutador conectada a un terminal, el proceso de notificaciones y asignaciones de *SAQs* se desarrolla independientemente de la posición de la raíz, que puede detectarse en la salida de cualquier conmutador de la red (lo que incluye, por supuesto, a aquellas salidas no conectadas a terminales). No en vano, la información sobre la posición de la raíz, incluida en las notificaciones, indica una ruta expresa hasta un punto de la red, y no un identificador de terminal.

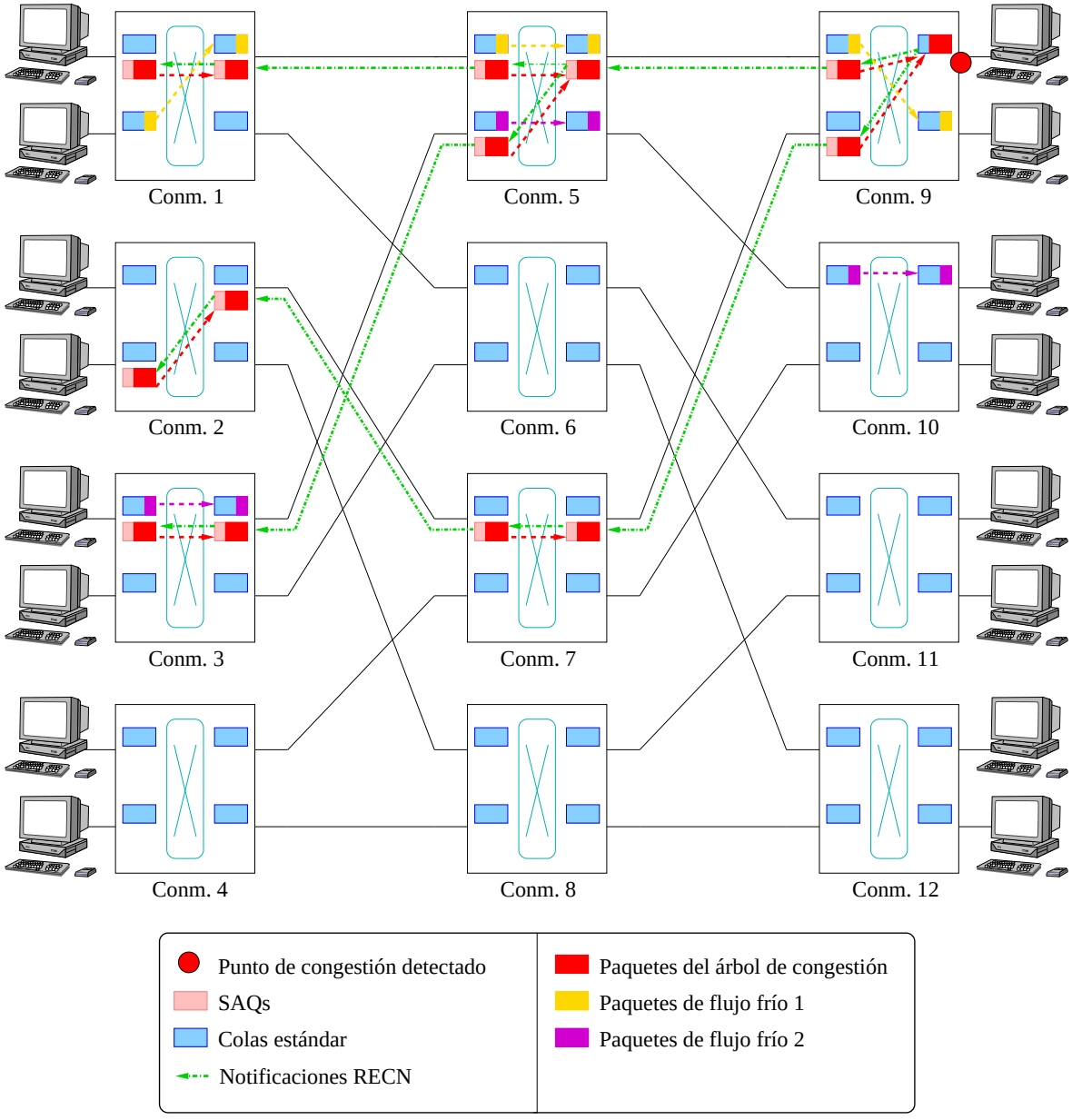


Figura 4.12: Propagación de la posición de la raíz de un árbol de congestión en una red completa, con la correspondiente asignación de *SAQs* en todas las ramas del árbol.

4.3.5. Gestión de múltiples árboles de congestión simultáneos

Cabe precisar que el proceso descrito para la propagación de la información sobre la posición de la raíz de un árbol de congestión es independiente de la existencia de otros árboles en la red. Así, distintas series de notificaciones, comunicando cada una de ellas la posición de diferentes puntos de congestión, pueden propagarse por la red simultáneamente y sin interferencias.

Como es lógico, si un puerto recibe notificaciones sobre distintos puntos de congestión, asignará *SAQs* diferentes a cada uno de los distintos puntos notificados, siempre y cuando disponga de suficientes *SAQs* libres. Recordemos que la única precaución necesaria en estas circunstancias es comparar la ruta de cada paquete que llegue al puerto con la asociada a cada una de las *SAQs* activas, y almacenarlo en la cola estándar únicamente si no hay ninguna coincidencia. Nótese que esta política supone no sólo separar los paquetes pertenecientes a flujos congestionados de los paquetes pertenecientes a flujos no congestionados, sino también separar los distintos flujos congestionados entre sí. Esto implica que, aun sin ser el objetivo principal del mecanismo, *RECN* también elimina el *HOL blocking* que los flujos congestionados pudieran producirse unos sobre otros.

Un caso que requiere especial tratamiento se produce cuando existen en un puerto distintas *SAQs* asignadas a árboles cuyas raíces se encuentran situadas en una misma dirección respecto al puerto, pero a distintas distancias del mismo. Es decir, las rutas asociadas a estas *SAQs* cumplen la propiedad de que, dadas dos de ellas cualesquiera, la ruta de menor longitud (que es la que lleva hasta la raíz más cercana de las dos) siempre está incluida en (es una subruta de) la ruta de mayor longitud (que es la que lleva hasta la raíz más lejana de las dos). En estos casos diremos que la *SAQ* asociada a la ruta de mayor longitud es **más específica** que la *SAQ* asociada a la ruta de menor longitud.

La figura 4.13 muestra una de las situaciones descritas en el párrafo anterior. En el caso representado, existe un puerto de entrada ($P3$ en el conmutador 1) con *SAQs* (*SAQ0*, *SAQ1* y *SAQ2*) asignadas a árboles cuyas raíces (raíces A, B y C) se sitúan "alineadas" a lo largo de un único camino que parte del puerto de entrada y llega hasta la raíz más alejada (raíz C). Observando la información de los registros *CAM* asociados a las *SAQs* mencionadas, puede observarse que *SAQ1*, asignada al árbol cuya raíz es raíz C, es más específica que el resto, pues está asociada a la ruta de mayor longitud (o, lo que es lo mismo, a la raíz más distante). Nótese que el resto de rutas (las asociadas a *SAQ0* y *SAQ2*) son subrutas de la ruta asociada a *SAQ1*. Siguiendo el mismo razonamiento, *SAQ0* es más específica que *SAQ2*.

El motivo por el que situaciones como la que refleja la figura suponen una complicación es la coincidencia múltiple que puede darse al comparar la ruta de un paquete que llegue al puerto con las rutas asociadas a todas las *SAQs*. En general, si la ruta de

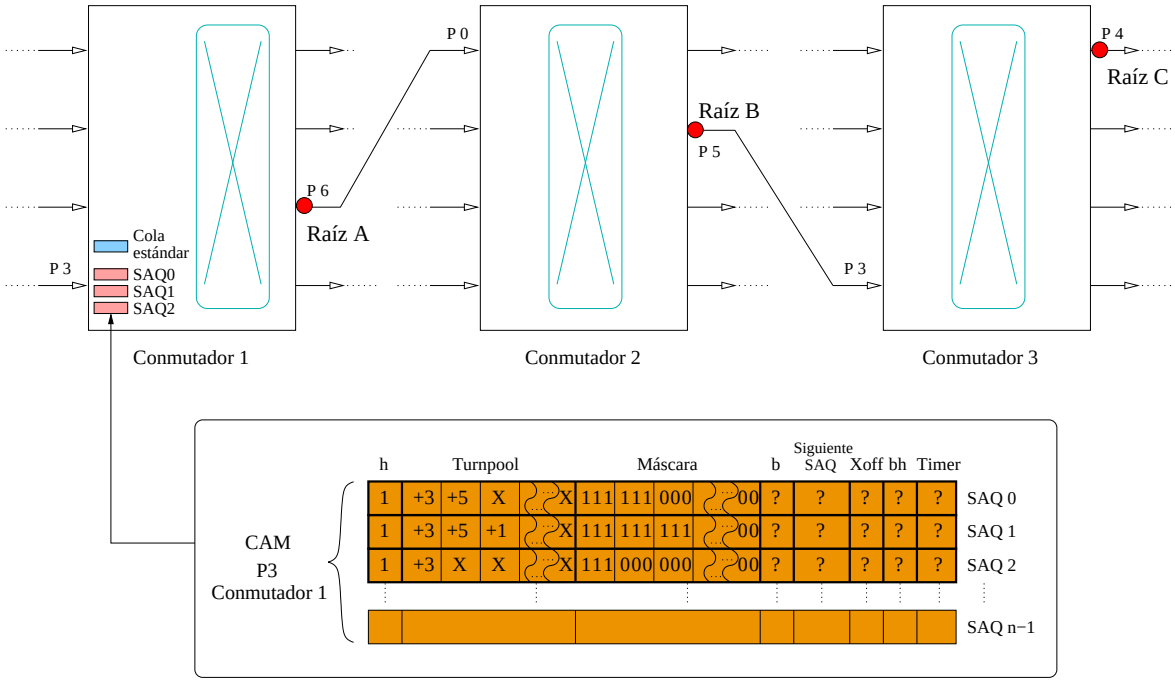


Figura 4.13: SAQs de un mismo puerto asignadas a raíces alineadas en un camino.

un paquete que llega a un puerto presenta una coincidencia con la ruta asociada a una *SAQ* de dicho puerto, también presentará coincidencia con las rutas asociadas a *SAQs* del mismo puerto que sean menos específicas que la primera, si estas *SAQs* existen. Por ejemplo, en la figura anterior, un paquete que llegue a $P3$ en el conmutador 1 y cuya ruta coincida con la asociada a *SAQ1*, también presentará una coincidencia al compararse con *SAQ2* y *SAQ0*. Nótese que esto debe ser forzosamente así, teniendo en cuenta que todos los paquetes que desde $P3$ se dirijan a la raíz C (asociada a *SAQ1*), también pasarán por las raíces A y B (asociadas a *SAQ2* y *SAQ0*, respectivamente).

En estas circunstancias, se plantea la incertidumbre sobre la *SAQ* donde debe almacenarse el paquete. Para resolver esta cuestión *RECN* establece que un paquete cuya ruta presenta múltiples coincidencias con rutas de distintas *SAQs* debe almacenarse en la *SAQ* más específica de todas ellas. Nótese que no se trata de una norma arbitraria: de otro modo, paquetes pertenecientes a distintos árboles de congestión se almacenarían en la misma cola, pudiéndose producir cierto *HOL blocking* entre ellos.

Así pues, es fundamental que la ruta de un paquete que llega a un puerto sea comparada con la ruta asociada a todas las *SAQs* activas en dicho puerto, incluso si se encuentra una coincidencia antes de haber terminado todas las comparaciones. Y si se encuentra coincidencia con la ruta asociada a una *SAQ*, sólo debe considerarse si no se habían encontrado antes otras coincidencias o si, habiéndolas encontrado, eran coincidencias con rutas de menor longitud (o sea, asociadas a *SAQs* menos específicas).

De este modo, cada paquete se almacenará en la *SAQ* que realmente corresponde al árbol de congestión al que pertenece el paquete.

4.3.6. Liberación de recursos

Una de las premisas fundamentales de *RECN* es que los recursos destinados a eliminar el *HOL blocking* se asignan y liberan de forma dinámica. Es decir, dado que una *SAQ* se asigna a un árbol de congestión cuando su existencia se pone de manifiesto, recíprocamente una *SAQ* debe liberarse cuando el árbol de congestión asociado desaparezca. Por supuesto, una *SAQ* liberada puede volver a asignarse a un nuevo árbol de congestión si es necesario. En consecuencia, una *SAQ* podrá ser utilizada en distintos momentos para almacenar paquetes de diferentes árboles de congestión mientras éstos no existan simultáneamente en la red (o, más exactamente, en el punto donde se sitúa la *SAQ*). Esta filosofía permite que un número limitado de *SAQs* por grupo sean suficientes para eliminar el *HOL blocking* que distintos árboles de congestión pudieran producir a lo largo del tiempo.

Para implementar la liberación de una *SAQ* será necesario desactivar el bit de habilitación de su registro *CAM* asociado, y (opcionalmente) borrar la información contenida en el resto de campos de dicho registro.

Evidentemente, es necesario establecer algún criterio para tener la completa seguridad de que una *SAQ* se liberará únicamente cuando su árbol de congestión asociado haya desaparecido realmente. Nótese que una liberación de *SAQs* "prematura" produciría el almacenamiento de los paquetes del árbol en colas estándar, con la consiguiente introducción de *HOL blocking*. Para evitarlo, *RECN* propone una liberación ordenada de las *SAQs*, de modo que, mientras que el proceso de asignación de sucesivas *SAQs* para un árbol se desarrolla desde la raíz del mismo hacia sus hojas, el proceso de liberación de *SAQs* se producirá desde las hojas a la raíz del árbol asociado.

Para implementar dicha liberación ordenada, *RECN* establece que una *SAQ* se liberará si y sólo si se cumplen simultáneamente las siguientes condiciones:

- **La *SAQ* está vacía.** Evidentemente, si una *SAQ* no está vacía, no puede considerarse que el árbol de congestión asociado haya desaparecido en el punto donde se sitúa la *SAQ*. La comprobación de esta condición es inmediata teniendo en cuenta que en todo momento hemos asumido que puede conocerse el nivel de ocupación de cualquier *buffer*.
- **La *SAQ* está situada en una hoja del árbol.** Si la *SAQ* no estuviera en una hoja del árbol, existirían puntos situados en sentido *upstream* por donde circulen paquetes del árbol (almacenados en *SAQs*, en teoría), y por tanto dicho árbol no puede darse por desaparecido. Nótese que incluso estando una *SAQ* vacía en un

momento dado, si no es una hoja del árbol podría volver a llenarse con paquetes del árbol enviados desde *SAQs* situadas en sentido *upstream*. Recuérdese que una *SAQ* es consciente de ser o no una hoja del árbol gracias al bit de hoja (*SAQs* en entradas) o a la lista de *tokens* (*SAQs* en salidas) de su registro *CAM* asociado, por lo que puede comprobarse fácilmente si esta condición se cumple o no.

- **Ha transcurrido un tiempo mínimo dado desde que la *SAQ* fue asignada.** Esta condición es necesaria para evitar que una *SAQ* se libere justo después de su asignación, ya que desde ese instante y hasta que reciba un paquete, la *SAQ* será con toda certeza una hoja del árbol y también estará vacía, cumpliendo las dos primeras condiciones. Esta última condición se implementa por medio del *timer* del registro *CAM* asociado a la *SAQ*. Como ya se ha mencionado, cuando la *SAQ* es asignada, su *timer* se inicia con un valor inicial (TD, *Time to Deallocate*), y se decrementa a cada instante hasta valer cero. El valor del *timer* se comprueba cuando se cumplen las otras condiciones para la liberación, que sólo se producirá si dicho valor es cero. Es decir, mientras el valor del *timer* sea mayor que cero, la *SAQ* no se liberará.

Teniendo en cuenta que se pretende una liberación ordenada de todas las *SAQs* asignadas a un árbol, desde las hojas a la raíz, y considerando también la segunda de las condiciones expuestas, se hace imprescindible que una *SAQ* sea consciente de volver a estar en una hoja del árbol, lo que sucederá tras liberarse todas las *SAQs* del árbol situadas en sentido *upstream* respecto a ella. Es decir, debe implementarse un mecanismo que actualice el bit de hoja (o la lista de *tokens*) del registro *CAM* asociado a una *SAQ* cuando el proceso de liberación haya llegado a todas las *SAQs* inmediatas en sentido *upstream*.

Para cubrir esta necesidad, *RECN* establece que una *SAQ* que se libera debe comunicar este hecho a la *SAQ* situada inmediatamente en sentido *downstream* que esté asignada al mismo árbol de congestión. Esto es, la comunicación de liberación debe enviarse a la *SAQ* que aparece en el campo "siguiente *SAQ*" del registro *CAM* asociado a la *SAQ* que se libera.

El efecto producido por una comunicación de liberación depende de la posición en el conmutador de las *SAQs* implicadas. Una comunicación de liberación desde una *SAQ* de salida en un conmutador a una *SAQ* de entrada en el conmutador "*downstream*" siguiente producirá que, inmediatamente, la *SAQ* de entrada receptora de la comunicación pueda activar su correspondiente bit de hoja, ya que su única *SAQ* inmediata en sentido *upstream* es la *SAQ* de salida emisora. Esto implica que una sola comunicación de liberación basta para que una *SAQ* cumpla la segunda condición para liberarse, y así lo hará automáticamente si se cumplen el resto de condiciones.

La figura 4.14 representa un ejemplo de liberación de una *SAQ* de salida (*SAQ2* de *P6* en el conmutador 1), junto con las correspondientes consecuencias. Así, puede

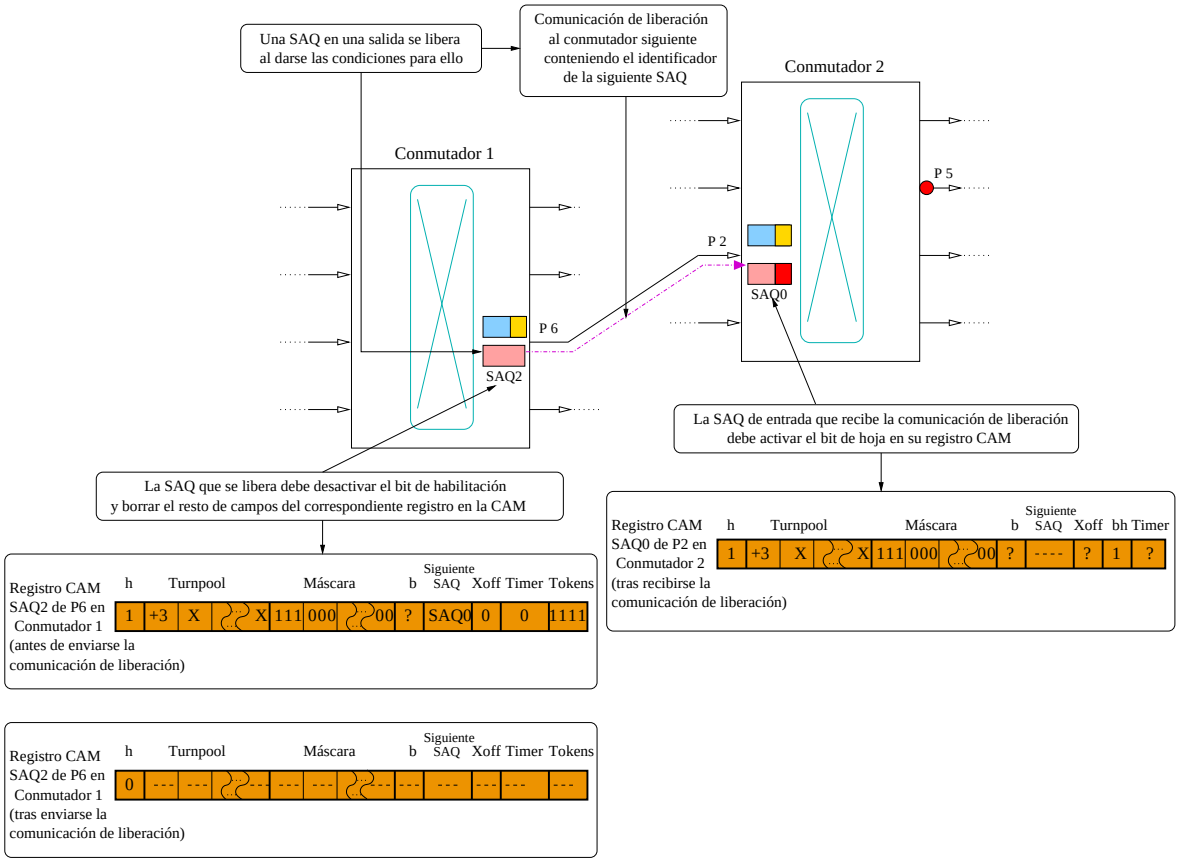


Figura 4.14: Comunicación de liberación de una *SAQ* de salida a una *SAQ* de entrada.

observarse el envío de la comunicación de liberación hacia la siguiente *SAQ downstream* en la entrada de otro conmutador (*SAQ0* en *P2* en el conmutador 2). También se reflejan los ajustes que se realizan en los registros *CAM* asociados a las *SAQs* implicadas. Nótese que la información contenida en el campo "siguiente *SAQ*" del registro asociado a la *SAQ* de salida debe incluirse en la comunicación de liberación, por lo que el contenido de este registro no debe borrarse hasta después de que dicha comunicación se envíe.

En el caso de liberación de *SAQs* situadas en una entrada de un conmutador, la correspondiente comunicación debe enviarse hacia una salida del mismo conmutador. El cálculo de la salida de destino de esta comunicación es trivial si consideramos que toda *SAQ* es consciente de la ruta hasta la raíz del árbol de congestión asociado, y por tanto, el primer salto de la ruta asociada a una *SAQ* de entrada lleva hasta la salida inmediata en sentido *downstream*, en el mismo conmutador, por donde circula el árbol. En dicha salida habrá una *SAQ* asignada al mismo árbol, salvo en el caso de que se trate del conmutador donde se sitúa la raíz del árbol, en cuyo caso las *SAQs* asociadas al árbol en las entradas se han asignado como consecuencia de una notificación directa de detección procedente de una cola estándar. Esto no plantea ninguna incertidumbre,

pues basta con consultar la información contenida en el campo "siguiente *SAQ*" del registro *CAM* correspondiente a la *SAQ* de entrada que se libera para saber si existe o no una *SAQ downstream*. En cualquier caso, haya una *SAQ* o una cola estándar en sentido *downstream*, la *SAQ* de entrada que se libera debe enviar una comunicación de liberación hacia la salida correspondiente.

Si una *SAQ* en una salida recibe una comunicación de liberación, hay que considerar que esta *SAQ*, a diferencia de una *SAQ* de entrada, no puede considerarse automáticamente situada en una hoja del árbol. Esto es debido a que una *SAQ* de salida puede haber comunicado congestión a varias entradas, y, por tanto, sólo cuando todas las *SAQs* asignadas en estas entradas se liberen, la *SAQ* de salida estará en una hoja del árbol. Por tanto, una *SAQ* de salida sólo se liberará tras recibir comunicaciones de liberación desde todas las entradas a las que haya notificado congestión. Recordemos que una *SAQ* de salida controla las entradas ya notificadas mediante su lista de *tokens*, y, en consecuencia, el efecto de recibir una comunicación de liberación en una *SAQ* de salida debe ser el que se reactive en dicha lista el *token* correspondiente a la entrada emisora de la comunicación. De este modo, una *SAQ* de salida volverá a estar en una hoja del árbol cuando todos los *tokens* de la lista se hayan reactivado, pudiendo en ese momento la *SAQ* de salida liberarse a su vez si se cumplen el resto de condiciones.

El proceso de liberación de *SAQs* se detiene al llegar a las *SAQs* situadas en las entradas del conmutador donde se encuentra la raíz del árbol. Sin embargo, y como ya se ha mencionado, también en este caso debe mandarse una comunicación de liberación precisamente a la salida congestionada que es la raíz. Esta comunicación será recibida por la cola estándar, y su efecto será similar al producido en una *SAQ* de salida. En este caso, la recepción de una comunicación de liberación supondrá que se desmarque el *token* correspondiente a la entrada emisora en la lista de *tokens* raíz asociada a la cola estándar. Nótese que esto es necesario por si dicha salida volviera a congestionarse: si el correspondiente *token* no se desmarcara al liberarse una *SAQ* de entrada, la cola estándar no comunicaría la nueva congestión a dicha entrada.

Puede comprobarse que el proceso descrito de liberación de *SAQs* asignadas a un árbol de congestión se desarrollará, como se pretendía, de forma ordenada y desde las hojas del árbol hacia la raíz del mismo, al progresar en este sentido las comunicaciones de liberación. Dicho de otro modo, la liberación de *SAQs* "avanza" en sentido *downstream*, al contrario que la asignación de *SAQs* para el árbol, que lo hace en sentido *upstream*.

4.3.7. Otros aspectos

En las secciones precedentes se han expuesto aquellos aspectos de *RECN* directamente orientados a eliminar el *HOL blocking*. Sin embargo, la implementación directa

del mecanismo tal y como se ha descrito, podría tener como consecuencia otros efectos indeseables, ante los que conviene tomar las precauciones pertinentes. Las siguientes secciones se ocupan de concretar tanto los mencionados efectos "colaterales" como las medidas adoptadas por *RECN* para evitarlos.

4.3.7.1. Control de flujo

Como se expuso en la sección 4.2.3, *RECN* asume que todas las colas activas en un puerto en un momento dado se implementan por medio de una única *RAM* que es compartida por dichas colas de forma dinámica. Por tanto, no existen límites, a priori, en el tamaño que pueda ocupar una cola mientras éste no sobrepase el total disponible en la *RAM*.

Es perfectamente posible, en este entorno, asegurar que una *RAM* nunca se desbordará mediante, por ejemplo, un sistema convencional de control de flujo por créditos a nivel de puerto, mediante el cual los puertos emisores no enviarán a los receptores mientras no dispongan de créditos para ello (ver sección 2.3). Recordemos que los créditos disponibles para envío desde un puerto se inicializan con un valor equivalente a la capacidad de la *RAM* del puerto receptor, disminuyen con cada paquete enviado, y aumentan cuando el puerto receptor "devuelve" créditos tras emitir a su vez paquetes. *RECN* asume que este sistema de créditos a nivel de puerto se usa como control de flujo básico.

Ahora bien, en el caso de que la memoria del puerto se comparta dinámicamente por varias colas, el control de flujo por créditos a nivel de puerto, aun evitando que la *RAM* del puerto se desborde, no puede evitar por sí mismo que la ocupación de una cola pueda crecer hasta ocupar toda la *RAM*. Esto podría suceder perfectamente con una *SAQ* en caso de almacenar un flujo congestionado particularmente intenso (constante o a ráfagas), lo que dejaría sin espacio en el puerto a otras *SAQs* o la cola estándar. Nótese que, en estas circunstancias, puede introducirse *HOL blocking* desde el puerto con la *RAM* agotada hacia las colas estándar situadas en otros puertos en sentido *upstream*, al no existir ningún espacio disponible para almacenar los paquetes que dichas colas quisieran enviar.

Para solucionar este problema, *RECN* propone limitar la ocupación máxima que puede alcanzar una *SAQ*. Para ello, se establece un control de flujo entre *SAQs* del tipo basado en marcas (o sea, modelo *Stop & Go*, ver sección 2.3), adoptándose para este control de flujo "especial" la denominación *Xon/Xoff*, como en *Advanced Switching*[4] (ver sección 2.7.3).

[4] Nótese que, al estar este control de flujo contemplado en la especificación *Advanced Switching*, su uso por parte de *RECN* no introduce una complicación adicional de cara a una implementación del mecanismo en redes *AS*.

Para implementar este control de flujo especial, toda *SAQ* cuya ocupación alcance cierto nivel (nivel de *Xoff*) enviará un mensaje de control *Xoff* a aquella *SAQ upstream* que le envíe paquetes. A su vez, una *SAQ* que reciba un mensaje *Xoff* activará el bit de *Xoff* en su registro *CAM* asociado, y mientras dicho bit esté activo, la *SAQ* no podrá enviar paquetes.

Nótese que el nivel de *Xoff* debe situarse por encima del umbral de notificación de congestión, ya que, obviamente, en caso contrario se enviarían mensajes *Xoff* a *SAQs* todavía no asignadas. Aunque este nivel corresponderá a una ocupación alta, debe considerarse que hasta que el mensaje de *Xoff* surta efecto, pueden seguir recibiéndose mensajes, con lo cual no conviene que este nivel se sitúe demasiado cerca de la ocupación máxima que se desea para la *SAQ*.

Por otra parte, el mensaje de *Xoff* debe contener el identificador de la *SAQ* emisora, para determinar en el puerto receptor, mediante una consulta a los campos "siguiente *SAQ*" de todos los registros de la *CAM*, qué *SAQ* del puerto debe considerarse la receptora del mensaje *Xoff*. Además, téngase en cuenta que si la *SAQ* que alcanza el nivel *Xoff* es una *SAQ* de salida, debe enviar mensajes (internos) de *Xoff* sólo hacia aquellos puertos a los que previamente haya notificado congestión (es decir, hacia aquellos puertos donde existen *SAQs* asignadas al mismo árbol que la *SAQ* emisora).

Así pues, una *SAQ* que haya enviado un mensaje *Xoff* debería dejar de recibir paquetes, con lo que su nivel de ocupación descenderá al ir enviando los paquetes que contiene. Cuando el nivel de ocupación de una *SAQ* que haya enviado un mensaje de *Xoff* descienda hasta cierto nivel (nivel de *Xon*), la *SAQ* enviará un mensaje de *Xon* a aquella *SAQ upstream* a la que se hubiera enviado el mensaje de *Xoff* anteriormente. Lógicamente, una *SAQ* que reciba un mensaje de *Xon* desactivará el bit de *Xoff* de su registro *CAM* asociado, y por tanto desde ese momento podrá volver a enviar paquetes.

En cuanto al nivel de *Xon*, debe corresponder a una ocupación baja, pero no tanto como para producir que la *SAQ* que lo emite sufra inanición. Más concretamente, debe tenerse en cuenta que durante el tiempo que tarde el mensaje de *Xon* en llegar y surtir efecto, y el que tarden en llegar nuevos paquetes a la *SAQ*, se seguirán emitiendo paquetes desde ésta, por lo que este nivel no debe situarse demasiado cerca de la ocupación nula. Nótese que, teniendo en cuenta lo expuesto para los niveles de *Xoff* y *Xon*, el umbral de notificación de congestión se situará entre ambos niveles. En consecuencia, cuando se asigna una *SAQ* y se rellenan los campos del registro *CAM* correspondiente, el valor inicial del bit de *Xoff* será 0 (desactivado).

En la figura 4.15 puede observarse un ejemplo elemental del funcionamiento del control de flujo *Xon/Xoff* entre *SAQs*. Se aprecia cómo la *SAQ* de entrada (*SAQ0* en *P*2 en el conmutador 2) que llega al nivel *Xoff* envía inmediatamente un mensaje de *Xoff* en sentido *upstream*, sin necesidad de incluir en él ninguna información salvo su propio identificador. Nótese que es innecesario enviar información sobre la ruta

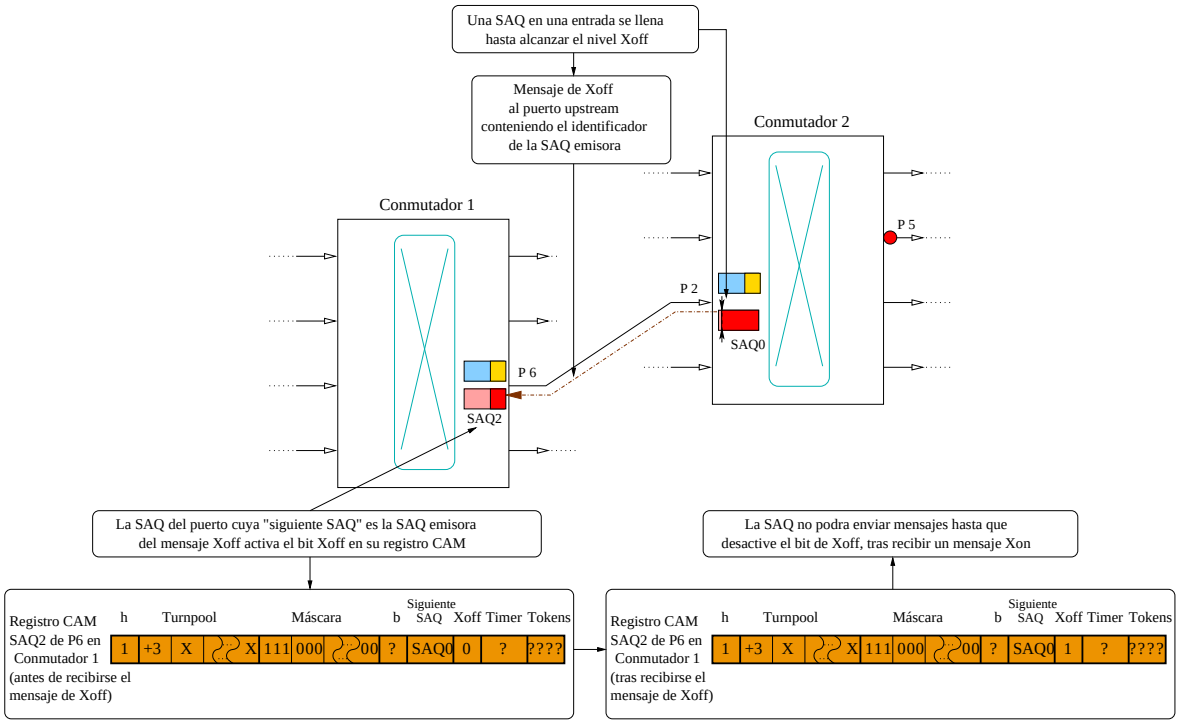


Figura 4.15: Control de flujo *Xon/Xoff* entre *SAQs* asignadas a un mismo árbol.

asignada a *SAQ0* (en este caso, la ruta hasta $P5$), puesto que el puerto de salida receptor del mensaje *Xoff* ($P6$ en el conmutador 1) sabe que la *SAQ* de dicho puerto que se corresponde con la *SAQ0 downstream* es la *SAQ2*, pues así lo indica el campo "siguiente *SAQ*" de esta última. En consecuencia, se activa el bit de *Xoff* del registro *CAM* asociado a *SAQ2*, que no podrá enviar paquetes a la *SAQ0 downstream* hasta que no reciba de ésta un mensaje *Xon*.

En definitiva, *RECN* asume que existirán dos tipos de controles de flujo operando a distintos niveles: por una parte un control de flujo basado en créditos a nivel de puerto, que afectará a todos los envíos de paquetes, y por otra un control de flujo *Xon/Xoff* que afectará sólo a los envíos de paquetes entre *SAQs*.

4.3.7.2. Garantía de entrega de paquetes en orden

La mayoría de las actuales tecnologías de redes de interconexión contemplan la garantía de que los paquetes de una transmisión lleguen a su destino en el mismo orden en que fueron emitidos. Esto es debido a los requisitos planteados por ciertos entornos o aplicaciones, como por ejemplo aquéllos destinados al manejo de tráfico multimedia.

Mientras que es sencillo garantizar la entrega en orden de los paquetes en redes basadas en conmutadores con una única cola *FIFO* en sus puertos y que usen encaminamiento determinista, no puede decirse lo mismo, a priori, para redes que usen encaminamiento adaptativo o para aquéllas cuyos conmutadores dispongan de varias colas en los puertos para almacenar los paquetes, incluso si éstas siguen una política *FIFO*. En el caso de existir múltiples colas en los puertos, no puede garantizarse directamente la entrega en orden si paquetes con una misma ruta (origen a destino) pueden almacenarse simultáneamente en colas distintas de un mismo puerto. Si esto se permite, y dependiendo del arbitraje, podría darse prioridad a la transmisión de paquetes que han llegado al puerto con posterioridad a otros que, siguiendo la misma ruta, se encuentran almacenados en otra cola, lo que desordenaría el flujo de paquetes que siguen dicha ruta.

Teniendo en cuenta lo expuesto, *RECN* podría introducir desorden en los flujos al poderse almacenar a la vez en la cola estándar y en alguna SAQ paquetes de un mismo flujo. Nótese que esto sucederá siempre que, al asignarse una SAQ en un puerto, la cola estándar contenga ya paquetes cuya ruta pasa por el punto de congestión asociado a esta nueva SAQ activa. Si esto es así, una vez asignada la SAQ, todos los paquetes que lleguen al puerto y sigan la ruta asociada a la misma, se almacenarán en la SAQ, y por tanto existirán en el puerto paquetes que siguen una misma ruta pero se encuentran almacenados en distintas colas.

Una posible solución a este problema consistiría en mover a la SAQ, justo tras su asignación, todos aquellos paquetes que están almacenados en la cola estándar pero que podrían almacenarse en la SAQ (por supuesto, respetando en ésta el orden relativo de estos paquetes en la cola estándar). Sin embargo, esta solución plantea el grave inconveniente de requerir el acceso a todos y cada uno de los paquetes almacenados en la cola estándar para comprobar su ruta. Y, teniendo en cuenta que todas las colas del puerto se implementan en una *RAM*, cuyo tiempo de acceso no es despreciable, no es difícil advertir que este proceso ralentizaría enormemente el mecanismo en sí y el tráfico en general.

En consecuencia, se necesita una solución alternativa a la expuesta, y para ello *RECN* propone lo siguiente: una SAQ no podrá enviar paquetes hasta que salgan todos los paquetes contenidos en la cola estándar en el momento de la asignación de la SAQ[5]. Evidentemente, esto garantiza que si la cola estándar contiene paquetes que siguen rutas que pasan por el punto de congestión asociado a la SAQ, todos ellos se enviarán antes que los que se almacenen en la SAQ tras su asignación, respetándose así el orden de llegada al puerto. En el caso de que la cola estándar no contuviera ningún

[5]Nótese que esta restricción se aplica independientemente de si existen o no en la cola estándar paquetes que potencialmente podrían ocasionar entrega fuera de orden. Esta decisión se debe a que detectar la presencia o ausencia de estos paquetes implicaría examinar la ruta de todos los paquetes contenidos en la cola estándar cada vez que se active una SAQ en el puerto, lo que complicaría en exceso la implementación del mecanismo e introduciría retardos adicionales.

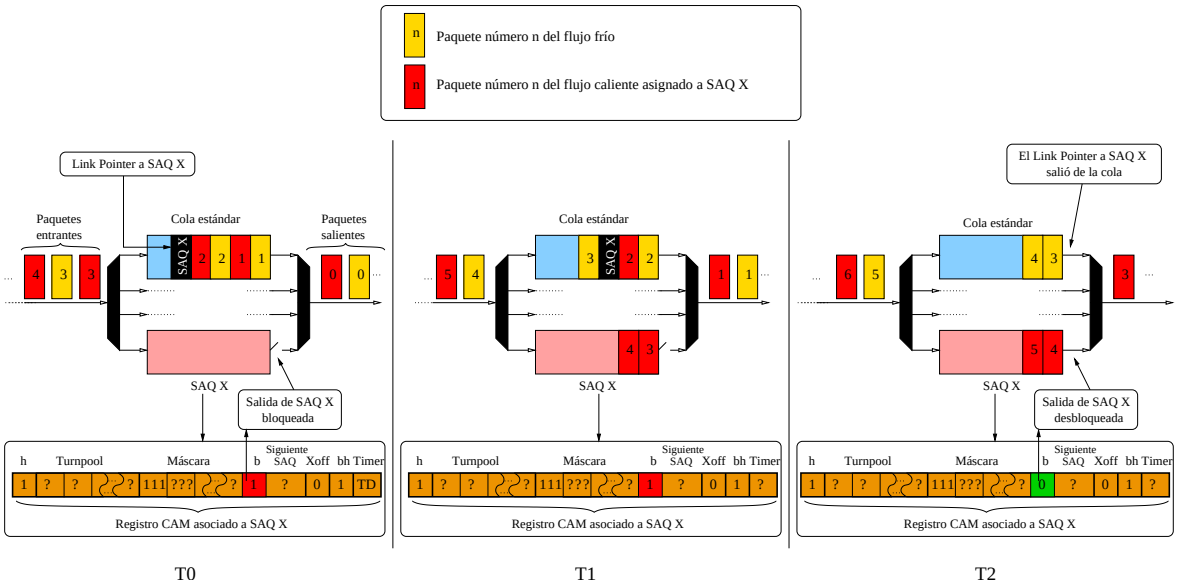


Figura 4.16: Tránsito de paquetes por un puerto, en distintos instantes, tras la asignación de una *SAQ* a la que apunta un *link pointer*.

paquete en el momento de la asignación de una *SAQ*, ésta podría enviar paquetes tan pronto como recibiera alguno.

Para implementar esta restricción en las *SAQs* se empleará el bit de bloqueo del registro *CAM* asociado a cada *SAQ*, de modo que mientras este bit permanezca activo en un registro *CAM*, la *SAQ* asociada se considerará bloqueada para enviar paquetes. Así, al asignarse una *SAQ* en un puerto, el bit de bloqueo se activará automáticamente si la cola estándar en dicho puerto contiene algún paquete, y se desactivará en caso contrario. Obviamente, si la cola estándar contiene paquetes en el momento de la asignación de una *SAQ*, debe existir un mecanismo que desactive el bit de bloqueo correspondiente cuando el último de estos paquetes sea enviado. Para ello, en el momento de asignarse una *SAQ*, se colocará en la cola estándar, justo tras el último paquete, una marca (*link pointer*) a modo de puntero a la *SAQ* asignada. Cuando un *link pointer* avance hasta la cabeza de la cola estándar significará que todos los paquetes precedentes han sido enviados, y por tanto la *SAQ* a la que apunta dicho *link pointer* ya puede desbloquearse, para lo cual se desactivará automáticamente el correspondiente bit de bloqueo.

En la figura 4.16 se ha representado la situación y el contenido de las colas de un puerto en tres instantes consecutivos ($T0$, $T1$ y $T2$), inmediatamente posteriores a la asignación de una *SAQ* (*SAQX*), para la que también se muestra su registro *CAM* asociado. La figura también incluye, para cada instante, los paquetes que están a punto de llegar al puerto y los que acaban de ser enviados desde el mismo. Los paquetes representados pertenecen a dos flujos: un flujo frío y un flujo congestionado,

para el que se asigna la *SAQX*. Cada paquete se distingue del resto de paquetes de su flujo mediante un número que indica su posición en el flujo.

En el ejemplo mostrado se supone que en el instante $T0$ la *SAQX* acaba de asignarse, y por ello está vacía. No sucede lo mismo en ese instante con la cola estándar, por lo cual se coloca en ella, tras el último paquete, un *link pointer* a *SAQX* y se activa el bit de bloqueo en el registro *CAM* de *SAQX*. Después de ese instante, aunque *SAQX* recibe paquetes del flujo congestionado, no puede emitirlos mientras el bit de bloqueo esté activo, como sucede en el instante $T1$. En dicho instante puede comprobarse que esto evita que un paquete (paquete 3 del flujo congestionado, en *SAQX*) pueda ser enviado antes que un paquete anterior en dicho flujo (paquete 2, en la cola estándar). Finalmente, cuando el *link pointer* "sale" de la cola estándar, se desactiva el bit de bloqueo del registro *CAM* asociado a *SAQX*, que ya puede enviar paquetes normalmente, como sucede en el instante $T2$.

Téngase en cuenta que una cola estándar podría contener perfectamente varios *link pointers*, apuntando a diferentes *SAQs*, si éstas se van asignando antes de que los *link pointers* que apuntan a las asignadas anteriormente hayan salido de la cola. Por supuesto, aunque coincidan temporalmente en la cola estándar, cada *link pointer* es totalmente independiente del resto.

Sin embargo, el sistema de *link pointers* descrito en los párrafos anteriores no evita todos los posibles casos en los que *RECN* puede introducir desorden en un flujo de paquetes. Este problema también puede surgir si se asigna en un puerto una *SAQ* más específica (ver sección 4.3.5) que otra *SAQ* ya asignada en dicho puerto. Si esto es así, la *SAQ* menos específica podría contener paquetes a los que, tras la nueva asignación, les correspondería ser almacenados en la *SAQ* más específica. En estas circunstancias, y siguiendo un razonamiento similar al expuesto para la cola estándar y las *SAQs*, se podrían enviar desde el puerto, desordenadamente, los paquetes del flujo correspondiente a la *SAQ* más específica.

Para solucionar estas situaciones, *RECN* establece que no debe aceptarse en un puerto una notificación de congestión que suponga la asignación de una *SAQ* más específica que otra *SAQ* ya activa en el puerto (por extensión, y de ahora en adelante, nos referiremos a estas notificaciones como "notificaciones más específicas"). Nótese que esto implica que en este puerto, la *SAQ* menos específica almacenará los paquetes que le corresponden realmente y también los paquetes que corresponderían a la *SAQ* "más específica" que no se asigna, con lo cual se introducirá cierto *HOL blocking*, al permitir que dos flujos congestionados distintos compartan cola. Pero debe tenerse en cuenta que se trata del *HOL blocking* de un flujo congestionado sobre otro (los flujos no congestionados nunca se verán afectados), y además ambos flujos comparten parcialmente una misma ruta, por lo cual el impacto de este *HOL blocking* en las prestaciones globales de la red no debería ser significativo.

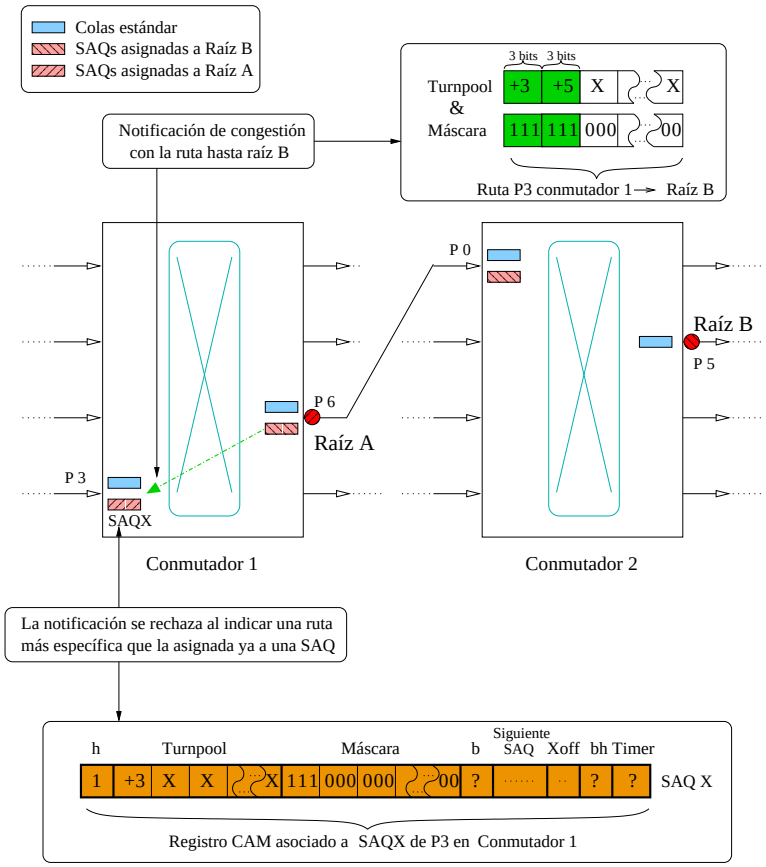


Figura 4.17: Rechazo en un puerto de una notificación de congestión más específica que una SAQ ya existente.

Un ejemplo de rechazo de notificación de congestión por el motivo que acabamos de explicar aparece en la figura 4.17. En ella, puede observarse que existen dos puntos de congestión detectados (raíces A y B), para los cuales se asignan $SAQs$ normalmente. Sin embargo, cuando la información sobre la existencia de la raíz B se propaga al puerto $P3$ del conmutador 1, ya existe en dicho puerto una SAQ ($SAQX$) asignada a una ruta menos específica (la que lleva hasta la raíz A) que la notificada. En estas condiciones, la notificación más específica es rechazada, y no se asignan $SAQs$ en el puerto $P3$ del conmutador 1 para la raíz B.

El cumplimiento de la restricción expuesta no implica que en un puerto no puedan existir $SAQs$ más y menos específicas, ya que las notificaciones que lleguen a un puerto y resulten ser menos específicas que una SAQ ya activa en dicho puerto sí serán atendidas, asignándose una nueva SAQ para la ruta menos específica indicada. Téngase en cuenta que esto no supone un riesgo de desorden, ya que, debido al criterio de almacenar paquetes siempre en la cola más específica para la que hay coincidencia, en la nueva SAQ (la menos específica) no se almacenará ningún paquete al que le corresponda estar

en una *SAQ* más específica, y a su vez, ésta no contendrá paquetes previos a los que les correspondería almacenarse en la nueva *SAQ*.

En definitiva, tomando las precauciones indicadas, *RECN* no introducirá desorden en ningún flujo de paquetes, sea éste o no un flujo congestionado.

4.4. Evaluación de la nueva propuesta

Habiendo descrito ya el planteamiento y los detalles operativos de *RECN*, es hora de confirmar mediante pruebas si la técnica, que, en teoría y según lo expuesto en las secciones precedentes, debe eliminar el *HOL blocking* de forma eficiente y escalable, realmente consigue este objetivo primordial.

En consecuencia, en las siguientes secciones se expone cómo se ha llevado a cabo la evaluación de *RECN*, y se muestran y analizan los resultados de dicha evaluación.

4.4.1. Método de evaluación

Como se ha comentado, en la evaluación pretendemos verificar si *RECN* es una técnica de eliminación del *HOL blocking* realmente eficiente y escalable a la vez. Nótese que ambas cualidades son independientes una de otra, por lo que sería necesario evaluar cada una de ellas por separado. Además, hay que tener en cuenta que el funcionamiento de *RECN*, tal como se ha indicado, depende del valor de ciertos parámetros específicos del mecanismo. Concretamente, tanto el nivel de los umbrales de detección y control de flujo *Xon/Xoff* como el número de *SAQs* por grupo que se empleen influirán en las prestaciones obtenidas. Por tanto, la evaluación de la eficacia y escalabilidad del mecanismo debería realizarse para los valores idóneos de los mencionados parámetros, con lo cual una fase previa de ajuste de dichos parámetros parece imprescindible.

En definitiva, podemos dividir la evaluación en tres análisis complementarios, que serían los siguientes:

1. **Un ajuste de los parámetros críticos del mecanismo.** Se trataría de encontrar (aunque fuera de forma aproximada) los valores de los distintos umbrales y del número de *SAQs* por puerto para los cuales *RECN* presenta un mejor comportamiento.

2. **Un estudio comparado de la eficiencia del mecanismo.** En este caso se trataría de comprobar hasta qué punto *RECN* elimina efectivamente el *HOL blocking* en situaciones de congestión. Este estudio sólo cobra pleno sentido si se

realiza de forma comparativa con el grado de eficiencia de otras técnicas de eliminación del *HOL blocking*, como por ejemplo la familia *Virtual Output Queuing* (*VOQ* a nivel de red y *VOQ* a nivel de conmutador).

3. **Un análisis de la escalabilidad del mecanismo.** Es decir, se trataría de comprobar que la eficiencia de *RECN* es realmente independiente del tamaño de la red, de modo que éste no condiciona el número de recursos requeridos por *RECN* para eliminar el *HOL blocking*.

Para realizar estos estudios con el mayor rigor posible, es preciso obtener, para distintas topologías de red y patrones de tráfico, y para distintas técnicas de eliminación o reducción del *HOL blocking* (incluyendo a *RECN*, por supuesto) resultados de métricas que nos indiquen tanto el grado de eliminación del *HOL blocking* que se consigue en cada caso como la cantidad y el grado de utilización de recursos de red empleados por el mecanismo.

Siguiendo la clasificación establecida en [Jai91], los resultados en los que se basa cualquier evaluación como la que nos ocupa, de sistemas o técnicas en el ámbito de los computadores, pueden obtenerse mediante diferentes métodos:

- **Mediciones reales.** Este método requiere que se realicen mediciones sobre un sistema real, ya implementado, que puede ser el sistema completo que se evalúa o un simple prototipo del mismo. En cualquier caso, se requeriría una implementación previa de la propuesta, a mayor o menor nivel. En el caso de *RECN*, el mecanismo se ha diseñado en todo momento para facilitar su aplicación en redes comerciales, por lo que su implementación es asequible, pero, en general, desde un punto de vista industrial. Además, la cantidad de elementos de red necesaria para obtener resultados en topologías de medio o gran tamaño excede de largo los recursos habitualmente disponibles en el mundo académico. En definitiva, la evaluación mediante este método se encuentra más allá de nuestras posibilidades.

- **Modelos analíticos.** También es posible realizar la evaluación basándose en un modelo analítico del sistema. Se trata de una solución asequible y sencilla, aunque esta última ventaja también constituye un gran inconveniente. Esto es debido a que una evaluación de este tipo sólo es posible si el modelo analítico refleja una versión simplificada del sistema o técnica a evaluar, lo que puede originar una pérdida de precisión en los resultados obtenidos. Puesto que una simplificación de *RECN* desvirtuaría casi con toda seguridad la medida de sus prestaciones, y deseamos el máximo rigor y precisión posibles en la evaluación, su realización mediante este método queda también descartada.

- **Simulaciones.** La simulación por computador está ampliamente aceptada hoy en día como un método de gran fiabilidad, y como tal es empleada en muy diversos ámbitos del mundo investigador. Este método permite modelar mediante un

programa (simulador), con un alto grado de precisión, cualquier sistema o técnica (y no sólo en el ámbito de los computadores). En general, los sistemas modelados mediante simuladores son aquéllos cuya complejidad escapa a las posibilidades de los modelos analíticos, que no pueden aportar el nivel de detalle que sí pueden contemplar los simuladores. Otra ventaja de la evaluación por simulación es que suele ser un método bastante asequible, requiriendo básicamente el esfuerzo de la programación del simulador y cierto número de computadores para ejecutar las pruebas. Por otra parte, tradicionalmente, el principal inconveniente de la evaluación mediante simulación era el elevado tiempo de ejecución necesario para completar las pruebas en evaluaciones de cierta complejidad. Pero en los últimos tiempos, el incremento en la velocidad de los procesadores, unido al uso cada vez más extendido de la computación paralela, han permitido reducir notablemente los tiempos necesarios para obtener resultados. Teniendo en cuenta lo expuesto, se ha considerado la simulación por computador como el método más conveniente para obtener resultados con los que evaluar *RECN*.

En definitiva, los resultados que se presentan en esta memoria para la evaluación de *RECN* se han obtenido utilizando un simulador, que hemos desarrollado expresamente para ello, y que modela el comportamiento de distintas redes de interconexión. En este tipo de simuladores es fundamental definir el modelo de carga de las pruebas, que representa el tráfico que se inyectará en las redes modeladas para la obtención de las distintas métricas. En general, las principales opciones que existen para modelar la carga son las siguientes:

- **Carga sintética**. En este caso, el tráfico que circula por la red es generado artificialmente, empleando una serie de funciones que, a partir de ciertos parámetros configurables, establecen en qué momento, con qué origen, con qué destino, y con qué tamaño se genera cada nuevo mensaje. Estas funciones, en general, reciben el nombre de **funciones de distribución** (de destinos, temporal, etc.), y muchas de ellas incluyen algún factor aleatorio. Variando las funciones de distribución, y los parámetros empleados, pueden generarse muy diversos patrones de tráfico con los que cargar la red, lo que permite la evaluación de distintas situaciones. Se trata de una opción muy utilizada, al ser un tipo de carga sencilla de definir y rápida de obtener, aunque presenta el inconveniente de no modelar en ocasiones el trafico real que circularía por la red en el sistema definitivo.

- **Trazas**. Los **ficheros de trazas** contienen la información del tráfico generado por aplicaciones en sistemas reales. Para generar estos ficheros, se realizan mediciones en un sistema real sobre el que se ejecuta una aplicación, de modo que se obtiene información sobre el destino y tamaño de los mensajes enviados, junto con el tiempo en el que fueron generados y otros datos que se consideren oportunos. La información contenida en un fichero de traza puede ser así "leída" por un

simulador para ser usada como carga con la que simular un tráfico real. Esta opción presenta la ventaja de permitir inyectar, de forma sencilla, tráfico real al sistema modelado. Su principal inconveniente es que el tráfico recogido en la traza es el resultado de ejecutar una aplicación concreta en un sistema concreto, por lo que podría no ser representativo del tráfico que correspondería a la ejecución de otras aplicaciones en el sistema evaluado. Además, no existen demasiados ficheros de trazas disponibles que recojan el tráfico en una red de interconexión, sobre todo si se trata de redes multiusuario donde debe respetarse el anonimato de las comunicaciones establecidas.

- **Carga real de aplicaciones**. Esta opción es más compleja que las anteriores, y consiste en ejecutar realmente una aplicación sobre el sistema simulado. En este caso la carga es generada directamente en el tiempo de ejecución del simulador por la propia aplicación. Los simuladores que siguen esta filosofía reciben el nombre de simuladores **dirigidos por ejecución**. Aunque es una opción muy útil cuando se pretende conocer la respuesta del sistema simulado ante la ejecución de una determinada aplicación, no lo es tanto para el caso de evaluación de sistemas donde pueden ejecutarse aplicaciones de muy distinta índole. Además, el coste computacional de este tipo de simulaciones es elevadísimo, sobre todo cuando se evalúan sistemas con un conjunto de parámetros medio o alto, donde, para una correcta evaluación, debe ejecutarse una aplicación paralela de cierta complejidad.

Teniendo en cuenta las ventajas e inconvenientes de las opciones descritas, hemos considerado la carga sintética y las trazas como las más convenientes para realizar la evaluación de *RECN*. La carga real de aplicaciones queda descartada por su elevado coste computacional y por la limitación que supone ceñirse a la carga de una determinada aplicación, considerando que el entorno simulado (una red de interconexión) está abierto a la ejecución de muy diversas aplicaciones. Por otra parte conviene aclarar que sólo disponemos de trazas para un número reducido de topologías, por lo que no puede recurrirse a este tipo de carga en todas las pruebas. En cuanto a la carga sintética, el inconveniente de no representar tráfico real puede superarse usando diversos patrones de tráfico, de los que resultan especialmente interesantes aquéllos que representan situaciones en las que puede producirse congestión.

Antes de mostrar los resultados obtenidos en los tres distintos análisis en los que hemos basado la evaluación, en la siguiente sección se detallan las características generales del simulador empleado en las pruebas. Las configuraciones concretas de las pruebas realizadas en cada análisis (topologías, cargas de tráfico, anchos de banda, etc.) se detallarán en las secciones correspondientes a cada uno de dichos análisis.

4.4.2. Herramienta de simulación

El simulador utilizado para realizar las distintas pruebas de la evaluación fue desarrollado a propósito para el presente estudio, aunque con las mismas características y estructura que otros simuladores con los que se llevaron a cabo estudios anteriores [Fli01, San02]. Para su programación se empleó el lenguaje Módula-2, lo que permitió el uso de algunos módulos comunes a los simuladores mencionados.

Se trata de un simulador basado en eventos, con una resolución de ciclo de reloj. La equivalencia temporal de cada ciclo de reloj puede ajustarse mediante un parámetro de entrada al simulador, lo que permite una gran flexibilidad para establecer distintas resoluciones temporales en función de la tecnología concreta simulada. Los tiempos asignados a los eventos modelados (medidos en ciclos de reloj) pueden seleccionarse mediante parámetros de entrada. En este sentido, hemos tenido la suerte de contar con la colaboración de la empresa [Ltd], que nos ha proporcionado los tiempos que corresponderían aproximadamente a cada evento en una red real.

Desde el punto de vista operativo, en cada ejecución del simulador debe indicarse el valor de una serie de parámetros de entrada. Los valores establecidos para cada prueba se han agrupado en un fichero de configuración que se ha utilizado como entrada en la ejecución de dicha prueba. En función de los parámetros de entrada introducidos, el simulador puede modelar distintas configuraciones de red, patrones de tráfico y mecanismos de control de congestión. Dicho modelado, con todas las opciones posibles, se detalla en los puntos inmediatamente posteriores.

Por otra parte, para cada prueba realizada, el simulador ofrece una amplia variedad de resultados reflejando distintas métricas significativas (productividad de la red, latencia de los paquetes, etc.). Las distintas métricas concretas que se han considerado en la evaluación se indican en el último punto de esta sección.

4.4.2.1. Modelado de la carga de tráfico

Como se ha comentado, en la evaluación de *RECN* emplearemos dos tipos de tráfico: patrones de tráfico sintético y trazas. El simulador permite seleccionar mediante un parámetro si se emplea uno u otro tipo de carga. En cualquier caso, la información de tráfico se incluye en un fichero de carga cuyo nombre debe indicarse al simulador (como parámetro de entrada) para que éste extraiga los datos. A continuación se describe el formato de la información contenida en estos ficheros:

- **Ficheros de carga sintética.** Si se selecciona la opción de carga sintética, el fichero de carga debe contener una línea (registro) por cada terminal de origen en

la red. En cada registro, tras el identificador del terminal, se indica el tipo de distribución de destinos de los mensajes que genera dicho terminal. Esta distribución puede ser:

- **Uniforme**: en este caso el destino de cada mensaje se generará de forma totalmente aleatoria. Si el registro correspondiente a un terminal indica esta distribución, a continuación debe indicarse la frecuencia (o tasa) de generación de mensajes en dicho terminal, como un porcentaje del ancho de banda del canal. Por último, el registro debe contener (en ciclos), tres valores que indican la duración de los intervalos de inactividad y actividad del terminal, y en qué instante cesa totalmente la generación de mensajes desde dicho terminal.
- ***Hot-spot***: si se indica esta distribución los paquetes generados por el terminal tendrán un destino preferente. En este caso, el registro debe contener, además de la frecuencia de generación de mensajes, el porcentaje de mensajes que se enviarán al destino preferente (el resto de mensajes se genera con destino aleatorio). A continuación debe indicarse el identificador del destino preferente. El registro concluirá, como en el caso de la distribución uniforme, con la duración de los intervalos de inactividad y actividad y con el tiempo final para la generación desde el terminal.

Variando en distintos ficheros las características del tráfico generado por los terminales pueden obtenerse los diferentes patrones de tráfico que nos interesen para la evaluación. Por ejemplo, si se establece que cierto número de terminales generen mensajes con distribución *hot-spot* a un mismo destino preferente, se creará casi con toda seguridad un árbol de congestión cuya raíz es dicho destino preferente. La intensidad de estos árboles también puede variarse cambiando el número de orígenes o su frecuencia de envío.

Un fichero de carga sintética es leído por el simulador completamente al comenzar la ejecución, y la información con las características de la generación de tráfico desde todos los terminales es almacenada en estructuras que permitirán calcular en qué momento y hacia qué destino se generarán los mensajes en cada terminal. En este caso, el tamaño de los mensajes se calcula de forma aleatoria, dentro de unos márgenes indicados como parámetros de entrada.

- **Ficheros de trazas.** Si se emplea un fichero de trazas el simulador leerá el fichero progresivamente durante su ejecución. Para ello, se asume que cada línea (registro) del fichero de trazas corresponde a un instante en que se genera un mensaje, y que dichos instantes están ordenados (aparecen en el fichero de trazas en orden creciente). Además del instante concreto de la generación, cada registro contiene el origen del mensaje, su destino y su tamaño en bytes. Estos datos son suficientes para simular la generación del mensaje en el instante asociado. Una

vez hecho esto, se lee el siguiente registro del fichero de trazas para programar la siguiente generación en el instante correspondiente, y así sucesivamente. Conviene recordar que los ficheros de trazas se generan por una aplicación real, por lo que en este caso no puede "dirigirse", como en el caso de carga sintética, la formación de árboles de congestión (lo cual no significa, por otra parte, que no se formen árboles).

En cualquier caso, e independientemente de en qué forma hayan sido generados, todos los mensajes son divididos en paquetes antes de ser inyectados a la red. El tamaño de paquete empleado en cada prueba es seleccionable mediante un parámetro del simulador.

4.4.2.2. Modelado de la red de interconexión

El simulador permite seleccionar el uso de redes de distintas topologías. En las pruebas hemos usado principalmente redes multietapa (patrón *perfect shuffle* [DYN02]) por ser la topología más comúnmente empleada hoy en día tanto en los *routers* con gran número de puertos como en los grandes *clusters* dedicados a computación paralela. Para comprobar la validez de *RECN* en otras topologías, también se han realizado pruebas con mallas 2-D. La selección de una u otra topología implica el uso de distintas funciones de interconexión en el simulador. Las dimensiones concretas de la red son configurables mediante parámetros de entrada al simulador. En cualquier caso, la red estará compuesta por un número variable de conmutadores y enlaces punto a punto, con un número también variable de terminales conectados. El modelado de cada tipo de componente de la red se detalla a continuación:

- **Enlaces:** se han modelado asumiendo que son bidireccionales, y que permiten transmisión *full-duplex* y segmentación de datos. El ancho de banda de los enlaces es seleccionable mediante un parámetro de entrada al simulador. En cuanto a la resolución de la transmisión de datos, se simula a nivel de paquete.

- **Conmutadores:** el modelado se basa en la arquitectura asumida por *RECN*, descrita en la sección 4.2.2. El número de terminales conectados a cada conmutador se selecciona mediante parámetro, por lo que el número total de puertos por conmutador es variable[6]. En estos puertos, tanto a la entrada como a la salida, el simulador modela una memoria *RAM* cuyo tamaño se indica también como parámetro. La gestión de estas memorias depende del mecanismo de control de congestión seleccionado para cada prueba (en este sentido, nos remitimos al punto dedicado al modelado de estos mecanismos). En cuanto al conmutador interno,

[6] Aún más teniendo en cuenta que en las redes multietapa puede variar el número de enlaces que conectan los conmutadores entre sí.

se ha asumido el uso de un *crossbar* multiplexado tanto a la entrada como a la salida, cuyo valor de *speedup* puede seleccionarse mediante un parámetro de entrada al simulador. La técnica de conmutación modelada es *virtual cut-through*. El árbitro que regula el acceso al crossbar es el descrito en la sección 4.2.2. Es decir, se atienden las peticiones de cruce según su antigüedad, y se da prioridad de cruce a las colas estándar sobre las *SAQs* (si las hubiere). En concreto, se ha modelado una prioridad 2 a 1: mientras haya paquetes para ello, cruzará un paquete desde una *SAQ* por cada dos desde colas estándar. Para regular el acceso de las colas de una misma salida al enlace correspondiente, se ha modelado un arbitraje con *round-robin* ponderado, con preferencia absoluta para las colas estándar sobre las *SAQs* (si las hubiere).

- **Terminales:** el simulador asume que se conectan a los conmutadores usando adaptadores de entrada (*Input Adapters*, *IAs*). Cada *IA* se modela con un número fijo de colas de admisión, siguiendo un esquema *VOQnet* (tantas colas como terminales en el sistema) y un número variable de colas de inyección, con el mismo esquema y tamaño que exista en los puertos de salida de los conmutadores (que a su vez depende de la técnica de control de congestión seleccionada). Por ejemplo, puede haber *SAQs* en las salidas de los *IAs* si se usa *RECN*. Cuando se genera un mensaje, se almacena en la cola de admisión del *IA* asignada a su destino, donde es dividido en paquetes que pasarán a una cola de inyección.

Por otra parte, el simulador permite usar distintos tipos de encaminamiento, seleccionables mediante parámetros de entrada. Se han realizado pruebas empleando tanto encaminamiento determinista como adaptativo:

- **Encaminamiento determinista:** la mayoría de pruebas se han realizado empleando esta opción, aunque con distintos algoritmos dependiendo de la topología de la red:

  - En mallas 2-D se ha empleado encaminamiento *dimension-order*, en concreto de tipo **X-Y** [DYN02].
  - En redes multietapa se ha empleado el encaminamiento ***self-routing*** [DYN02] propio del patrón de conexión *perfect shuffle*.

  Por otra parte, en el caso de encaminamiento determinista, la técnica de encaminamiento modelada por el simulador se ajusta totalmente al encaminamiento fuente empleado por *Advanced Switching*, de modo que cada paquete se modela con un *turnpool* y un *turn pointer* que indican el salto a realizar por el paquete en cada conmutador de su ruta. En estos casos, para cada red, todas las rutas, en forma de saltos, entre cualquier par origen-destino de terminales han sido generadas mediante una aplicación complementaria, y grabadas (una ruta para cada par

origen-destino) en un fichero de rutas que es leído por el simulador al comenzar su ejecución. Esto favorece la velocidad de ejecución de cada prueba, al no tener que calcularse constantemente las rutas de los paquetes.

- **Encaminamiento adaptativo:** este tipo de encaminamiento se ha considerado para la realización de algunas pruebas debido a que, hasta cierto punto, su uso puede reducir el *HOL blocking* (ver sección 3.1). En concreto, se ha empleado sólo con redes multietapa, con un criterio de escoger la salida con la cola más vacía en las etapas en las que el *self-routing* permite libertad de selección.

En cuanto al control de flujo, se asume que se realiza a nivel de paquete. El simulador modela distintas políticas de control de flujo, cuyo uso depende de la técnica de control de congestión empleada:

- Para *RECN*, se usa control de flujo basado en créditos a nivel de puerto, que pueden ser consumidos para transmitir paquetes desde una cola estándar o desde una *SAQ*. Internamente al conmutador, existe un control de flujo similar. En definitiva, el total de créditos disponibles para una entrada (o salida) depende del tamaño de la memoria total en la siguiente salida (o entrada). Además, se ha modelado el control de flujo especial *Xon/Xoff* que *RECN* establece entre *SAQs*. Los niveles concretos de *Xon* y *Xoff* empleados son ajustables mediante parámetros de entrada.

- Para los mecanismos de control de congestión basados en colas estáticas (todos salvo *RECN*) se ha modelado un control de flujo basado en créditos por cola. En este caso, el número de créditos disponible para una cola de entrada o salida depende del tamaño de la memoria en el siguiente puerto y del número de colas definidas en dicho puerto.

En cualquier caso, los mensajes de control de flujo que circulen por la red comparten el ancho de banda disponible en los enlaces con los paquetes de datos.

4.4.2.3. Modelado de las técnicas de control de congestión

El simulador modela distintos mecanismos de control de congestión, lo cual es imprescindible para poder evaluar las prestaciones de *RECN* respecto a las ofrecidas por otras técnicas. El uso de una u otra técnica es una opción del simulador, y se refleja principalmente en la forma en que se gestiona la memoria de los puertos para almacenar paquetes. A continuación se detalla el modelado de las distintas técnicas que pueden simularse:

- ***RECN*:** en este caso la memoria *RAM* de un puerto es compartida dinámicamente por la cola estándar y por todas las *SAQs* activas en dicho puerto en un momento dado. Debido al carácter dinámico de las *SAQs*, las celdas de memoria no se asignan a una cola a priori, sino que se asignan y se liberan sólo cuando es necesario y posible. Tanto para las entradas como para las salidas, puede seleccionarse un número de *SAQs* por grupo. Obviamente, se ha modelado la comparación de *turnpools* para determinar la cola donde debe almacenarse cada paquete. Además, se han modelado en el simulador el resto de detalles de *RECN*, incluyendo todos los mensajes de control que emplea el mecanismo (notificaciones, mensajes de control de flujo *Xon/Xoff*, etc.). Estos mensajes comparten el ancho de banda disponible en los enlaces con los paquetes de datos. El simulador acepta distintos valores para los umbrales de detección de congestión y de envío de notificaciones, así como para los niveles de *Xon* y *Xoff*.

- ***Virtual Output Queues* a nivel de red (*VOQnet*):** si se selecciona esta técnica, la memoria *RAM* del puerto de entrada o salida se divide equitativamente y de forma estática en tantas colas como terminales existan en total en el sistema. Es decir, las colas definidas sí tienen un tamaño definido a priori. Los paquetes se almacenan en la cola correspondiente a su destino. En definitiva, el parámetro que rige el modelado de la técnica es el que indica el número de terminales en la red.

- ***Virtual Output Queues* a nivel de conmutador (*VOQsw*):** en este caso la memoria *RAM* del puerto de entrada o salida se divide equitativamente y de forma estática (tamaño definido a priori) en tantas colas como puertos tenga un conmutador. Los paquetes se almacenan en la cola correspondiente a la salida que solicitan. Por tanto, el parámetro que rige el modelado de la técnica es el que indica el número de salidas en un conmutador.

- **Canales virtuales:** el número de canales virtuales del sistema es un parámetro de entrada al simulador. En estos casos la memoria *RAM* del puerto de entrada o salida se dividirá equitativamente y de forma estática en tantas colas como canales virtuales se indiquen. En este caso, para seleccionar la cola donde se almacenará un paquete se escoge aquélla que se encuentre más vacía (es decir, con un menor número de paquetes almacenados) en el momento de la llegada del paquete al puerto.

4.4.2.4. Métricas ofrecidas

Como ya se ha indicado, el simulador ofrece como resultado de cada simulación los valores de distintas métricas que típicamente se emplean en la evaluación del comportamiento de las redes de interconexión. El valor de cada una de estas métricas puede medirse en sucesivos instantes de tiempo a lo largo del periodo de simulación, por lo

que de cada simulación puede resultar un número dado de valores (muestras) para cada una de las métricas. El número de muestras en cada simulación puede variarse mediante un parámetro de entrada al simulador (que define concretamente el intervalo de muestreo).

La cantidad de métricas distintas ofrecidas por el simulador es considerable, por lo que a continuación nos limitaremos a describir exclusivamente las que se han tenido en cuenta en el presente estudio:

- **Productividad (*throughput*):** medida como la cantidad de bytes por unidad de tiempo que se recibe, en total, en todos los terminales conectados a la red durante un intervalo de tiempo determinado. Se trata de una métrica fundamental en la presente evaluación, pues la consecuencia más inmediata de la existencia de *HOL blocking* significativo en la red es un descenso severo en la productividad. Por tanto, la eficacia de una técnica de eliminación del *HOL blocking* puede valorarse a partir de la capacidad de dicha técnica para mantener la productividad durante situaciones de congestión. En las secciones donde se muestran resultados, la productividad obtenida en cada caso analizado se muestra bien en función del tiempo (en este caso se han tomado varias muestras de productividad a lo largo de cada simulación), bien en función de la tasa de generación de mensajes en los terminales (en este caso, para obtener los valores de productividad, se han realizado varias simulaciones, variando la mencionada tasa en cada una de ellas y recogiendo un único valor de productividad por simulación).

- **Latencia media:** tiempo medio que tardan los paquetes en llegar a su destino. El simulador ofrece resultados de latencia media medidos tanto desde la generación del paquete (latencia media desde la generación) como desde su inyección (latencia media de red). Al igual que la productividad, la latencia también aparecerá en los resultados en función tanto del tiempo como de la tasa de generación de mensajes en los terminales.

- **Número máximo de *SAQs* activas en un puerto:** esta métrica es exclusiva de las pruebas en las que se simule el empleo de *RECN* como técnica de control de congestión, e indica el número de *SAQs* que se encuentran activas en aquel puerto de la red que cuente con un mayor número de *SAQs* activas en un instante dado. El simulador distingue entre el número máximo de *SAQs* activas en puertos de entrada y el número máximo de *SAQs* activas en puertos de salida, por lo que realmente ofrece dos métricas distintas aunque similares.

- **Número total de *SAQs* activas en la red:** esta métrica también es exclusiva de las pruebas donde se emplee *RECN*, e indica el número total de *SAQs* activas en la red en un instante dado. Es decir, es la suma de las *SAQs* activas en todos los puertos de entrada o salida de la red en un instante dado.

- **Máximo porcentaje de uso de un enlace por mensajes de control:** se trata, para cada intervalo de muestreo, del máximo porcentaje de tiempo que un enlace de la red ha estado ocupado transmitiendo mensajes de control. Es decir, se considera este porcentaje para aquel enlace de la red donde haya existido un mayor tráfico de mensajes de control durante dicho intervalo.

4.4.3. Ajuste de los parámetros críticos del mecanismo

En este análisis se pretende descubrir los valores idóneos de aquellos parámetros específicos de *RECN* que pueden afectar directamente a sus prestaciones. Como ya se ha explicado, dichos parámetros son básicamente los siguientes:

- **Niveles de los umbrales de detección y control de flujo *Xon/Xoff*:** para el valor del umbral de detección, lo ideal sería encontrar un valor medio que no produjera detecciones de congestión prematuras (es decir, detecciones de situaciones de congestión que no son tales) ni detecciones de congestión tardías que llevaran a que *RECN* reaccionara demasiado tarde. En el primer caso, el efecto sería un consumo excesivo de *SAQs*, mientras que en el segundo, el efecto sería una incorrecta gestión de la congestión, ya que aparecería *HOL blocking* por la reacción lenta de *RECN*, con la consiguiente degradación de las prestaciones de la red.

  Respecto a los valores de los umbrales de *Xon* y *Xoff*, hay que tener en cuenta que ambos dependen del nivel del umbral de detección, pues por el propio funcionamiento del control de flujo sólo tienen sentido ciertos valores relativos de dichos umbrales. Así, el umbral de *Xoff* debe estar situado por encima del nivel del umbral de detección, ya que, en caso contrario, se intentaría poner en *Xoff* una *SAQ* para la que aún no se ha enviado notificación y que por tanto aún no se ha asignado. Por otra parte, el valor del umbral de *Xon* debería situarse por debajo del nivel del umbral de detección, ya que este último se asocia a situaciones de congestión significativas en un puerto, mientras que el primero está asociado a situaciones donde la congestión en dicho puerto está remitiendo. En definitiva, el ajuste de estos tres umbrales no es independiente, y por tanto realizaremos un único ajuste mediante pruebas con distintas ternas de valores que se ajusten a las posiciones relativas de estos umbrales que se acaban de explicar.

- **Número de *SAQs* por grupo:** respecto a este valor, hay que considerar que *RECN* no funcionará nunca peor al aumentar el número de *SAQs* por grupo. Ahora bien, teniendo en cuenta el planteamiento teórico del mecanismo, debería existir un valor del número de *SAQs* por grupo con el cual se obtendrían unas prestaciones tales (en cuanto a la eliminación del *HOL blocking*) que no mejorarían significativamente aun aumentando el número de *SAQs* por grupo.

Puesto que un objetivo prioritario en el diseño de*RECN* es conseguir las máximas prestaciones con el menor número de recursos posibles, el ajuste de este valor debe centrarse en encontrar este mínimo número de *SAQs* por grupo que permite gestionar correctamente las situaciones de congestión.

4.4.3.1. Configuración de las pruebas

Puesto que en este análisis inicial sólo se pretende descubrir las condiciones en las que *RECN* presenta un comportamiento óptimo, sólo se han realizado simulaciones empleando *RECN* como técnica de control de congestión, variando en las distintas pruebas los valores de los parámetros críticos del mecanismo ya mencionados.

En concreto, respecto al número de *SAQs* por grupo, se han realizado pruebas con 4, 8 y 16 *SAQs*. Respecto a los umbrales de detección, *Xon* y *Xoff*, hay que tener en cuenta que la memoria que se ha asumido en cada puerto de entrada o salida es de 32 KB, a partir de lo cual se han considerado para las distintas pruebas las ternas de valores (expresados en bytes) que se muestran en la tabla 4.1. Puede observarse que la posición relativa de los valores de cada terna es correcta según lo explicado anteriormente, y que los valores escogidos corresponden a configuraciones donde el valor del umbral de detección sería alto (terna 1), medio (terna 2) y bajo (terna 3). Concretamente, se ha considerado como valor alto del umbral de detección aquél que permitiría que se detectara congestión en cada una de las *SAQs* de un puerto estando ya la memoria del puerto prácticamente llena (teniendo en cuenta que puede haber hasta 16 *SAQs* por grupo y que se dispone de 32 KB). Los valores medio y bajo del umbral de detección corresponden respectivamente a la mitad y a la sexta parte del valor alto del umbral. En cada terna, los valores de los umbrales de *Xoff* y *Xon* se han situado respectivamente por encima y por debajo del valor del umbral de detección, y equidistan ambos de dicho valor en cierta cantidad de bytes. De la combinación de las tres ternas de valores de los umbrales con los tres valores indicados del número de *SAQs* por grupo resultan nueve configuraciones de los parámetros críticos de *RECN*, que serán las consideradas en el análisis.

Número de terna	Umbral de detección	Umbral de *Xoff*	Umbral de *Xon*
1	1920 bytes	2880 bytes	960 bytes
2	960 bytes	1280 bytes	640 bytes
3	320 bytes	448 bytes	192 bytes

Tabla 4.1: Valores de los umbrales de detección y control de flujo *Xon/Xoff* considerados en el ajuste de parámetros.

Dados los objetivos limitados de este análisis inicial, se ha considerado oportuno emplear en todas las pruebas un único modelo de red de interconexión, que se ajusta, en general, a las características de redes empleadas comúnmente en sistemas actuales.

		Tráfico uniforme			Tráfico árbol de congestión				
Caso de tráfico	Tamaño de red (BMIN)	Nº. de fuentes	Destino	Tasa de generación	Nº. de fuentes	Destino	Tasa de generación	Instante inicio	Instante fin
1	64×64	87,5 %	aleatorio	50 %	12,5 %	Terminal 32	100 %	800 μs	1100 μs
2	64×64	87,5 %	aleatorio	100 %	12,5 %	Terminal 32	100 %	800 μs	1100 μs
3	64×64	75 %	aleatorio	50 %	25 %	Terminal 32	100 %	800 μs	1100 μs
4	64×64	75 %	aleatorio	100 %	25 %	Terminal 32	100 %	800 μs	1100 μs
5	64×64	87,5 %	aleatorio	50 %	12,5 %	Terminal 32	100 %	800 μs	1400 μs
6	64×64	87,5 %	aleatorio	100 %	12,5 %	Terminal 32	100 %	800 μs	1400 μs
7	64×64	75 %	aleatorio	50 %	25 %	Terminal 32	100 %	800 μs	1400 μs
8	64×64	75 %	aleatorio	100 %	25 %	Terminal 32	100 %	800 μs	1400 μs

Tabla 4.2: Tráficos empleados en el ajuste de parámetros.

Así, en cuanto a la topología, se ha empleado una red multietapa bidireccional (*BMIN*) con 64 terminales conectados y con conmutadores de 8 puertos, de los cuales cuatro conectan al conmutador con otros de la etapa posterior y 4 con los de la etapa anterior (salvo en la primera etapa, donde 4 puertos conectan al conmutador a 4 terminales). Este esquema implica que la red tenga 3 etapas con 16 conmutadores por etapa). El ancho de banda efectivo de los enlaces se ha fijado en 8 *Gbps*[7]. Respecto a la memoria *RAM* de los puertos de los conmutadores, tal y como ya se ha indicado, se dispone de 32 KB en cada entrada o salida. Internamente, el *crossbar* de los conmutadores se ha configurado con un *speedup* de 1,5 (lo que, teniendo en cuenta el ancho de banda de los enlaces, implica un ancho de banda del *crossbar* de 12 *Gbps*).

En cuanto a los *Input Adapters* (*IAs*), como ya se ha comentado, sus colas de inyección siguen el mismo esquema que los puertos de salida de los conmutadores, por lo que también los *IAs* se han modelado con 32 KB de memoria disponible en su salida. Los *IAs* dividen los mensajes generados en paquetes de 64 bytes para inyectarlos en la red.

Respecto al tráfico empleado en las pruebas de este análisis, se ha escogido usar exclusivamente patrones de tráfico sintético que permiten generar en la red diversas situaciones de congestión que deberían ser resueltas por *RECN*, dejando el uso de trazas para análisis posteriores, con el ajuste de parámetros ya realizado. Los distintos casos de tráfico considerados se detallan en la tabla 4.2. En todos los casos existe un porcentaje de fuentes que inyecta tráfico uniformemente (es decir, con destino totalmente aleatorio) durante toda la simulación, mientras que el resto de fuentes se encarga de generar congestión inyectando intensamente tráfico a un único destino preferente durante un tiempo limitado. Puede apreciarse que para modelar los distintos casos, se han variado básicamente tres parámetros del tráfico: el porcentaje de fuentes que inyectan tráfico uniforme o congestionado, la tasa de generación de paquetes en las fuentes de tráfico uniforme y la duración del intervalo de actividad de las fuentes de tráfico congestionado.

[7]Cabe resaltar que este valor corresponde aproximadamente a un ancho de banda del tipo $\times 4$ de *PCI-Express* o *Advanced Switching*.

De este modo, los distintos casos se corresponden, como se pretendía, a situaciones con mayor o menor intensidad de tráfico (congestionado y no congestionado) en la red.

Evidentemente, se valorarán las distintas configuraciones de parámetros críticos de *RECN* en la medida en que sean válidas para eliminar el *HOL blocking* generado en las diferentes situaciones de tráfico. Es por ello que la principal métrica que se ha considerado en el análisis es la productividad de la red, obtenida para todas las combinaciones entre las configuraciones de parámetros críticos de *RECN* y los distintos casos de tráfico. Además, puesto que buscamos que esta eliminación del *HOL blocking* se realice con el menor número de recursos posible, también se ha considerado la cantidad de *SAQs* (totales y por puerto) requerida por cada una de estas combinaciones. A estas métricas y a estas combinaciones de tráficos y parámetros *RECN* corresponden por tanto los resultados que se muestran en la siguiente sección.

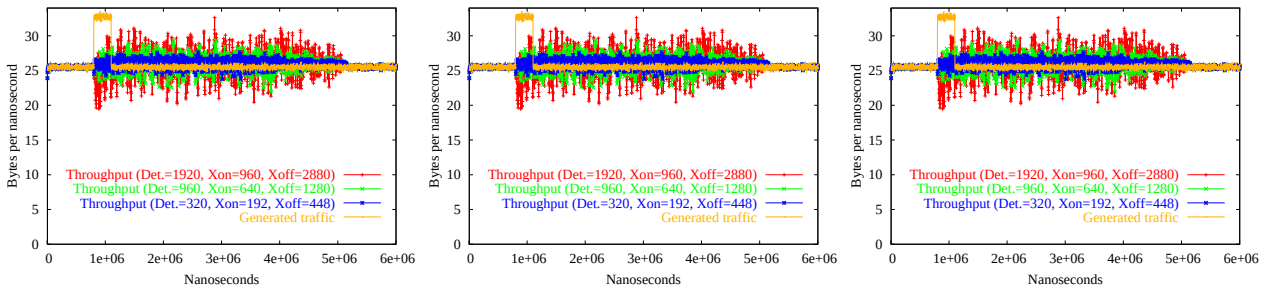
4.4.3.2. Resultados

Las figuras 4.18 y 4.19 muestran la productividad de la red en función del tiempo para cada caso de tráfico y cada configuración de parámetros críticos de *RECN*. Puede apreciarse que la figura 4.18 corresponde a aquellos casos de tráfico en que el intervalo de actividad de las fuentes de congestión es menor (tráficos 1 a 4), mientras que la figura 4.19 corresponde a los casos de tráfico con mayor duración de dicho intervalo (tráficos 5 a 8). También puede observarse que cada gráfica individual contiene resultados para un caso de tráfico y tres configuraciones de parámetros críticos (que corresponden a las tres distintas ternas de valores de umbrales junto a un único valor del número de *SAQs* por grupo). Por tanto, en cada fila de estas figuras (3 gráficas) aparecen todos los resultados para la combinación de un caso de tráfico con todas las configuraciones de parámetros críticos.

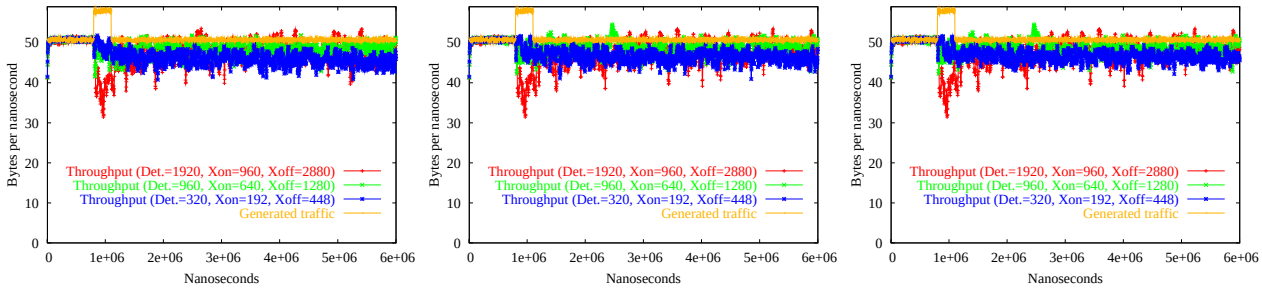
Por otra parte, cada gráfica individual contiene también una serie que representa el tráfico total generado en la red a lo largo del tiempo. Esta serie se incluye para apreciar en qué medida es eliminado el posible *HOL blocking* que surja en la red en los distintos casos, ya que cuanto más se acerque la productividad al tráfico generado (sin considerar el intervalo de inyección de tráfico congestionado) menos *HOL blocking* existe en la red.

Como puede verse en las figuras, en la mayoría de los casos la productividad no se aleja demasiado del tráfico inyectado, por lo que en estos casos puede decirse que el *HOL blocking* que pudiera producir la inyección de tráfico congestionado es eliminado de forma bastante satisfactoria.

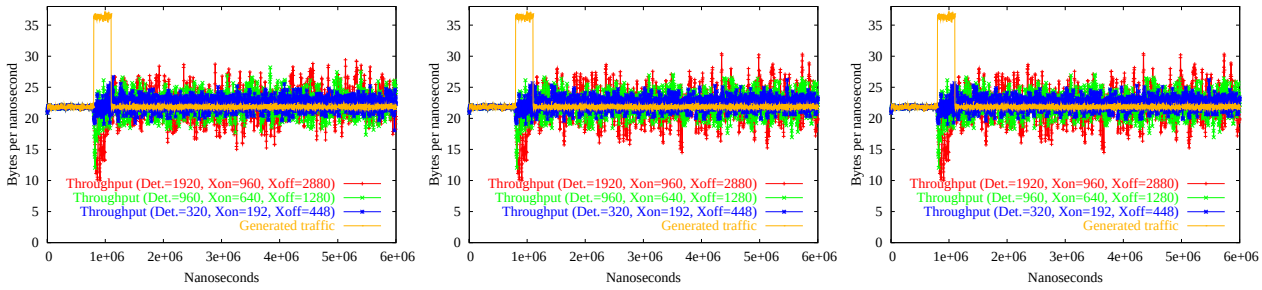
Sin embargo, existen algunos casos donde la productividad desciende notablemente para aquellas configuraciones de parámetros críticos con valores altos de los umbrales. Este efecto puede apreciarse especialmente en todas las gráficas correspondientes a los



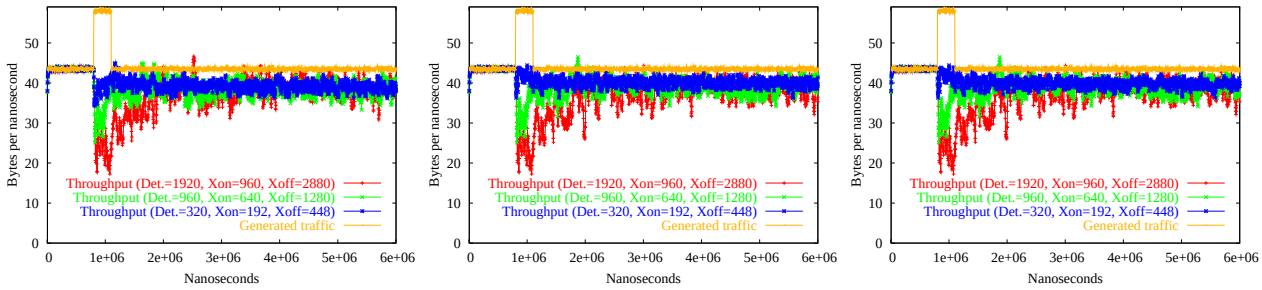
(a) Tráfico 1 - 4 SAQs (b) Tráfico 1 - 8 SAQs (c) Tráfico 1 - 16 SAQs



(d) Tráfico 2 - 4 SAQs (e) Tráfico 2 - 8 SAQs (f) Tráfico 2 - 16 SAQs

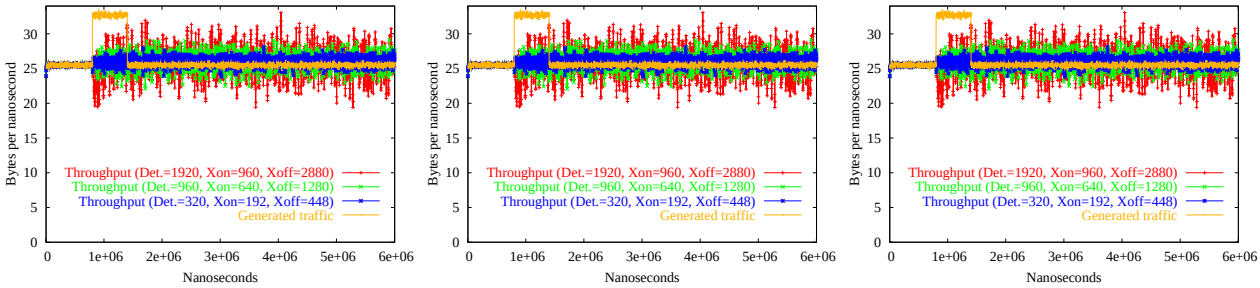


(g) Tráfico 3 - 4 SAQs (h) Tráfico 3 - 8 SAQs (i) Tráfico 3 - 16 SAQs

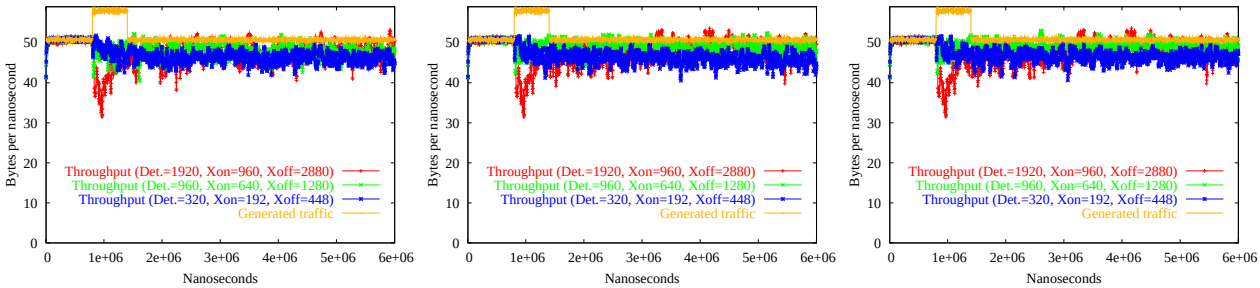


(j) Tráfico 4 - 4 SAQs (k) Tráfico 4 - 8 SAQs (l) Tráfico 4 - 16 SAQs

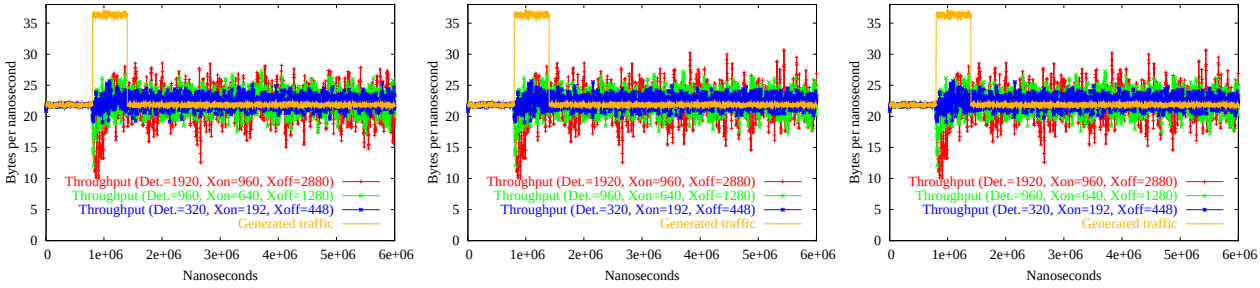
Figura 4.18: Productividad de la red en función del tiempo para las distintas configuraciones de parámetros *RECN* consideradas, y para los casos de tráfico 1 a 4.



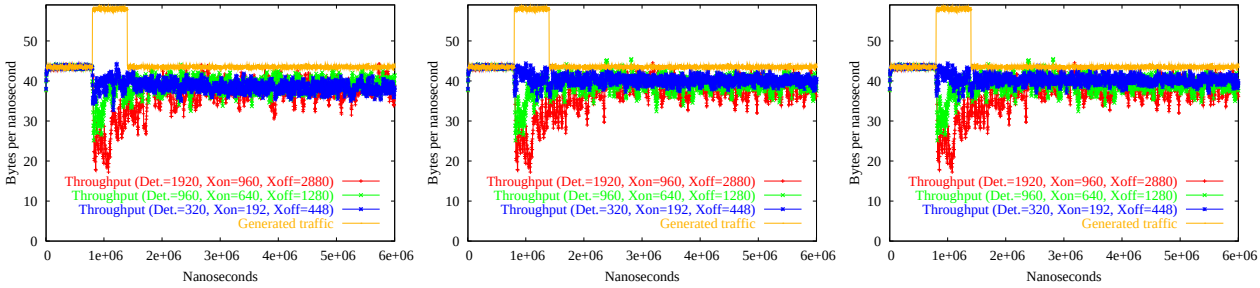
(a) Tráfico 5 - 4 SAQs (b) Tráfico 5 - 8 SAQs (c) Tráfico 5 - 16 SAQs



(d) Tráfico 6 - 4 SAQs (e) Tráfico 6 - 8 SAQs (f) Tráfico 6 - 16 SAQs



(g) Tráfico 7 - 4 SAQs (h) Tráfico 7 - 8 SAQs (i) Tráfico 7 - 16 SAQs



(j) Tráfico 8 - 4 SAQs (k) Tráfico 8 - 8 SAQs (l) Tráfico 8 - 16 SAQs

Figura 4.19: Productividad de la red en función del tiempo para las distintas configuraciones de parámetros *RECN* consideradas, y para los casos de tráfico 5 a 8.

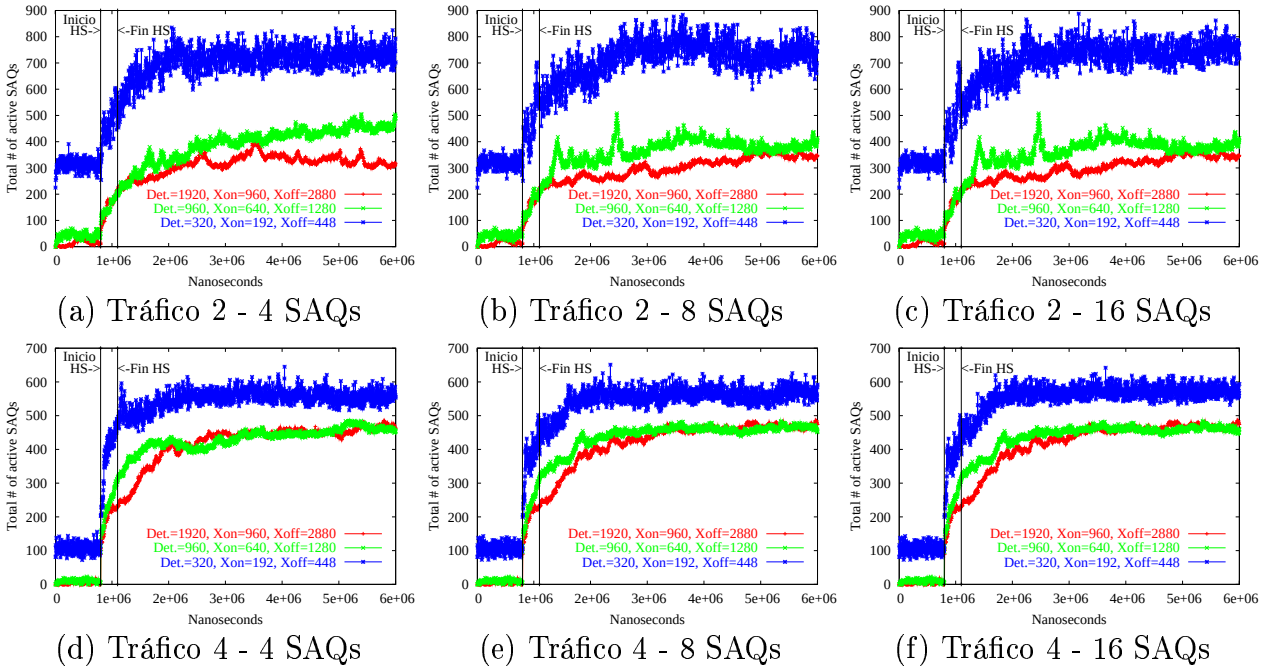
casos de tráfico 4 y 8 (4.18.j, 4.18.k, 4.18,l 4.19.j, 4.19.k y 4.19.l). Puede apreciarse que este descenso en la productividad no se produce sólo durante el intervalo de inyección de tráfico congestionado, sino que se prolonga más allá. Es decir, que en estos casos la productividad cae y tarda un tiempo apreciable en recuperar un valor cercano al tráfico inyectado. En menor medida, este efecto también se observa en el resto de gráficas de las figuras 4.18 y 4.19. Esto confirma la idea de que un valor demasiado alto del umbral de detección produce una detección tardía de la congestión, que provoca la aparición de cierto *HOL blocking* que acaba por afectar a la productividad. Es interesante destacar que este efecto es independiente del número de *SAQs* por grupo, lo cual es lógico pues el umbral de detección demasiado alto impide que se asignen todas las *SAQs* que debieran activarse para eliminar el *HOL blocking* (las *SAQs* están infrautilizadas). En definitiva, la terna de valores de umbrales altos parece descartable.

No está tan clara la elección entre las ternas de valores de umbrales medios y bajos. En algunos casos (tráficos 1, 3, 5 y 7, y para cualquier número de *SAQs* por grupo) ambas ternas consiguen resultados muy similares. Sin embargo, en todas las gráficas correspondientes a los tráficos 4 y 8, la terna de valores de umbrales bajos parece obtener resultados ligeramente mejores que la terna de umbrales medios, mientras que sucede lo contrario en las gráficas correspondientes a los tráficos 2 y 6[8]. Estas diferencias se explican teniendo en cuenta que, en los tráficos 2 y 6, el porcentaje de fuentes de tráfico uniforme es mayor, y además en ambos casos estas fuentes inyectan al 100 % de la capacidad del enlace. Por tanto, para estos tráficos, es fácil que la terna de umbrales bajos detecte como tráfico congestionado lo que no son sino ráfagas de tráfico uniforme hacia un mismo destino, llevando esto a una sobreexplotación de las *SAQs*, que se encontrarán en su mayoría ocupadas (y por tanto no disponibles) cuando comience a inyectarse tráfico congestionado. Por el contrario, los tráficos 4 y 8 son casos donde el número de fuentes de tráfico uniforme es menor, y por tanto, también es menor la probabilidad de "falsas" detecciones de congestión. Para estos tráficos, la terna de valores de umbrales bajos se beneficia de una mayor velocidad de detección de "auténticas" situaciones de congestión frente a la terna de umbrales medios.

En definitiva, parece que deba escogerse bien una configuración con probabilidad de falsas detecciones (terna de umbrales bajos), bien una configuración con menor rapidez para detectar auténticas situaciones de congestión (terna de umbrales medios).

Ante esta disyuntiva, recurriremos a resultados de otras métricas que aporten más datos al análisis. Por ejemplo, la figura 4.20 muestra el total de *SAQs* activas en la red a lo largo del tiempo de simulación, para todas las configuraciones de parámetros críticos de *RECN* y para los casos de tráfico 2 y 4. Se muestran los resultados sólo para estos

[8]Nótese que los resultados para los casos de tráfico 2 y 6 son virtualmente idénticos, y lo mismo sucede entre los casos 4 y 8. Esto implica que la duración del intervalo de inyección del tráfico congestionado no influye demasiado en las prestaciones obtenidas para las distintas configuraciones de parámetros.



(a) Tráfico 2 - 4 SAQs (b) Tráfico 2 - 8 SAQs (c) Tráfico 2 - 16 SAQs

(d) Tráfico 4 - 4 SAQs (e) Tráfico 4 - 8 SAQs (f) Tráfico 4 - 16 SAQs

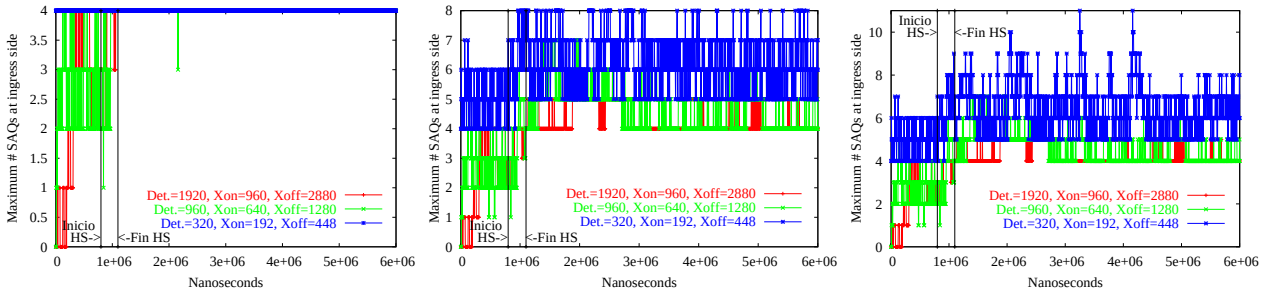
Figura 4.20: Número total de *SAQs* activas en la red en función del tiempo para las distintas configuraciones de parámetros *RECN* consideradas, y para los casos de tráfico 2 y 4.

casos de tráfico por ser aquéllos en los que los resultados de productividad presentan diferencias apreciables para las ternas de valores de umbrales medios y bajos[9]. A modo de ayuda, en cada gráfica individual se han marcado los instantes inicial y final del intervalo de inyección de tráfico congestionado.

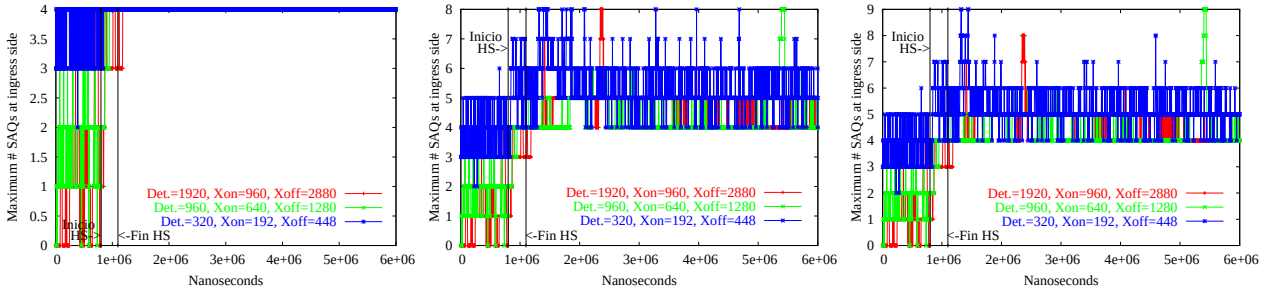
Como puede verse en la figura 4.20, en todos los casos de tráfico e independientemente del número de *SAQs* por grupo, la cantidad total de *SAQs* activas en la red es muchísimo mayor para la terna de valores de umbrales bajos. Esto es así incluso antes de que se produzca la inyección de tráfico congestionado, lo que confirma la idea de que un valor bajo del umbral de detección produce falsas detecciones de congestión.

Por el contrario, los resultados para la terna de valores de umbrales medios indican que el número de *SAQs* activas antes del comienzo de la inyección de tráfico congestionado es bastante reducido, por lo que con esta terna se produce una detección de la congestión más precisa. Teniendo en cuenta que el mantenimiento de un gran número de *SAQs* activas redundaría en un mayor consumo de energía y ciclos de procesamiento en el conmutador, y considerando que las diferencias entre los resultados de productividad de las ternas de valores de umbrales medios y bajos son bastante limitadas incluso en el peor caso, parece aconsejable decantarse por la terna de valores de umbrales medios.

[9]En cualquier caso, los resultados para otros casos de tráfico son cualitativamente similares a los mostrados.

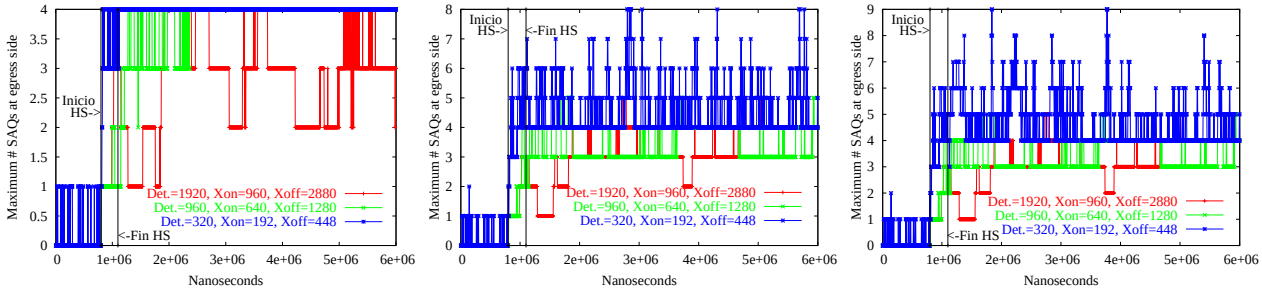


(a) Tráfico 2 - 4 SAQs (b) Tráfico 2 - 8 SAQs (c) Tráfico 2 - 16 SAQs

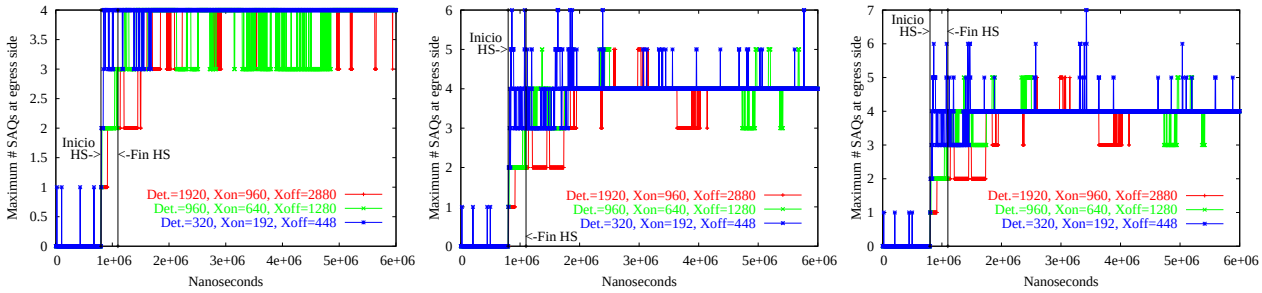


(d) Tráfico 4 - 4 SAQs (e) Tráfico 4 - 8 SAQs (f) Tráfico 4 - 16 SAQs

Figura 4.21: Número máximo de *SAQs* activas en una entrada en función del tiempo para las distintas configuraciones de parámetros *RECN* consideradas, y para los casos de tráfico 2 y 4.



(a) Tráfico 2 - 4 SAQs (b) Tráfico 2 - 8 SAQs (c) Tráfico 2 - 16 SAQs



(d) Tráfico 4 - 4 SAQs (e) Tráfico 4 - 8 SAQs (f) Tráfico 4 - 16 SAQs

Figura 4.22: Número máximo de *SAQs* activas en una salida en función del tiempo para las distintas configuraciones de parámetros *RECN* consideradas, y para los casos de tráfico 2 y 4.

Los resultados mostrados en las figuras 4.21 y 4.22 reafirman esta decisión. Ambas figuras muestran el máximo número de *SAQs* activas en una entrada (figura 4.21) o salida (figura 4.22) de un conmutador de la red a lo largo del tiempo de simulación, para todas las configuraciones de parámetros críticos de *RECN* y, como en la figura anterior, sólo para los casos de tráfico 2 y 4. También se ha representado como referencia en cada gráfica individual el inicio y el fin de la inyección de tráfico congestionado.

Como en la figura 4.20, estos resultados indican de nuevo que la terna de valores de umbrales bajos tiende a activar casi todas las *SAQs* de los puertos, sea o no realmente necesario (es decir, exista o no congestión real). Puede apreciarse que en algunos casos, el uso de esta terna lleva a mantener activas todas las *SAQs* disponibles en un grupo durante buena parte de la simulación. En definitiva, la cantidad máxima de *SAQs* activas para la terna de valores de umbrales bajos se sitúa notablemente por encima de la cantidad máxima para la terna de valores de umbrales medios, sin que ello suponga (a partir de los resultados de productividad) ninguna ventaja.

Habiendo seleccionado definitivamente la terna de valores de umbrales medios, aún podemos extraer otras conclusiones importantes de los resultados presentados en las figuras 4.21 y 4.22. En concreto, puede apreciarse que para la terna seleccionada, el número máximo de *SAQs* empleadas en una entrada o salida, considerando todos los resultados, es 9. Sin embargo, este resultado sólo se alcanza un breve instante de tiempo y para una única configuración, y, en general, el número de *SAQs* activas para esta terna se mantiene por debajo de 8 prácticamente en todos los casos y durante todo el tiempo de simulación. Es decir, 8 *SAQs* por grupo parecen suficientes para eliminar el *HOL blocking* empleando la terna de valores de umbrales medios. Por el contrario, en muchos instantes se mantienen activas para esta terna más de 4 *SAQs*, por lo que parece desaconsejable reducir a 4 el número máximo de *SAQs* por grupo. En definitiva, podemos fijar el número óptimo de *SAQs* por grupo en 8, lo cual no es en absoluto excesivo.

En conclusión, podemos cerrar la fase de ajuste de los parámetros críticos de *RECN* tras encontrar que los mejores resultados de *RECN* se obtienen para valores medios de los umbrales (siempre tomando como referencia el de detección), y que 8 *SAQs* por grupo son suficientes para el buen funcionamiento del mecanismo. Estos valores resultantes del ajuste han sido empleados para obtener los resultados de *RECN* empleados en los restantes análisis de la evaluación, a los que corresponden las siguientes secciones.

4.4.4. Comparación de prestaciones

El objetivo del presente análisis es evaluar las prestaciones obtenidas con *RECN* respecto a las obtenidas con otras técnicas de eliminación del *HOL blocking*. Dicho de otro modo, se trata de comparar, en idénticas condiciones de carga de tráfico y

estructura de la red, el distinto comportamiento del tráfico en la red cuando se use una u otra de estas técnicas.

Más concretamente, se ha considerado oportuno comparar *RECN* con las siguientes técnicas:

- ***Virtual Output Queues* a nivel de red (*VOQnet*):** Cabe recordar que con esta técnica, en teoría, se conseguirían excelentes prestaciones. De hecho, debería eliminar de forma ideal el *HOL blocking*, aunque a costa de utilizar gran cantidad de recursos. Por tanto, *RECN* no puede conseguir mejores prestaciones que esta técnica, de modo que consideraremos que las prestaciones obtenidas con *RECN* son buenas en la medida en que se aproximen a las obtenidas con *VOQnet*.

- ***Virtual Output Queues* a nivel de conmutador (*VOQsw*):** En este caso, el *HOL blocking* no se eliminaría totalmente, sino sólo el producido al nivel del propio conmutador, por lo que, al menos en teoría, *RECN* debería obtener mejores prestaciones.

- **Canales virtuales (*Virtual Channels*):** Como ya se indicó en la sección 3.3.2.3, esta técnica, aun no estando específicamente orientada a eliminar el *HOL blocking*, puede reducirlo en cierta medida. De nuevo, *RECN* debería presentar mejores prestaciones que esta técnica, puesto que con ella tampoco se elimina el *HOL blocking* completamente (salvo que se emplearan canales virtuales con configuraciones equivalentes a *VOQnet*, caso que consideramos incluido en la comparación con dicha técnica).

Además, y como referencia de cómo y cuánto pueden degradarse las prestaciones de la red en situaciones de congestión si no se usa ninguna técnica de eliminación del *HOL blocking*, también hemos obtenido resultados, en las mismas condiciones, para el caso de que sólo exista una única cola en cada puerto de los conmutadores, donde todos los paquetes entrantes al puerto se almacenarían indiscriminadamente. Evidentemente, cualquiera de las técnicas anteriormente mencionadas debería obtener mejores prestaciones que las obtenidas en este caso.

4.4.4.1. Configuración de las pruebas

El tipo de análisis que nos ocupa es conveniente realizarlo a partir de resultados obtenidos en una mayor variedad de entornos (respecto a las cargas de tráfico y la estructura de la red) que los considerados en el análisis anterior. De este modo, las prestaciones obtenidas con las distintas técnicas de control de congestión pueden compararse en escenarios de diversas características.

Por ejemplo, respecto al tráfico, además de patrones de tráfico sintético, también se han usado trazas. En concreto, disponemos de trazas proporcionadas por *Hewlett-Packard Labs*, generadas a partir de la actividad de entrada/salida medida desde el 14-1-1999 al 28-2-1999 en la interfaz de los discos del sistema *cello* [ssp]. Dicho sistema es un sistema de tiempo compartido con un subsistema de almacenamiento compuesto por 23 discos, de manera que estas trazas proporcionan información tanto de las peticiones generadas por los terminales conectados al sistema como de las respuestas generadas por los discos. Dada la antigüedad de estos datos, y considerando que la tecnología evoluciona rápidamente, permitiendo enlaces más rápidos y tiempos de acceso a disco menores, se ha considerado necesario aplicar distintos factores de compresión a las trazas. Mediante esta compresión, se divide entre cierto factor los tiempos entre generación de mensajes originalmente establecidos en la traza, aumentando así la frecuencia de generación de mensajes. Concretamente, hemos realizado pruebas con dos factores de compresión: 20 y 40.

En cuanto a los patrones de tráfico sintético, se han empleado algunos de los casos de tráfico ya utilizados en el análisis anterior, en concreto los casos 1 a 4 (pueden verse en la tabla 4.2). No se han considerado los casos de tráfico 5 a 8 debido a que, a la vista de los resultados de productividad de las figuras 4.18 y 4.19, la variación de la duración del intervalo de inyección de tráfico congestionado (que es la única diferencia entre estos casos y los casos 1 a 4) no parece influir demasiado en las prestaciones obtenidas. Independientemente del uso de trazas o tráfico sintético, los mensajes se dividen en paquetes de 64 bytes antes de ser inyectados.

Respecto a la estructura de la red, y teniendo en cuenta que también se realizará un posterior análisis sobre la escalabilidad de *RECN*, hemos optado por emplear redes de un único tamaño (*BMINs* de 64 terminales, como en el análisis anterior), dejando la pruebas con redes mayores para el siguiente análisis. Ahora bien, de cara a una mayor variedad, hemos realizado pruebas variando ligeramente las características de los conmutadores. En concreto, y considerando las limitaciones de los conmutadores reales en cuanto a *speedup*, hemos considerado en las pruebas tanto conmutadores con *speedup* de valor 1,5 como conmutadores con *speedup* de valor 1 (es decir, sin aceleración interna). Nótese que en este último caso, y teniendo en cuenta que mantenemos el ancho de banda de los enlaces en 8 *Gbps*, el ancho de banda interno del conmutador (es decir, del *crossbar*) es también de 8 *Gbps*. En cuanto al número de puertos de los conmutadores, hemos seguido empleando exclusivamente conmutadores con 8 puertos bidireccionales. Respecto al tamaño de la memoria de los puertos, mantenemos los 32 KB por entrada o salida empleados en las pruebas del análisis precedente.

Teniendo en cuenta las características del presente análisis, obviamente hemos realizado diversas simulaciones usando las distintas técnicas de control de congestión indicadas en el punto anterior. Para las simulaciones que usan *RECN*, hemos usado los

valores de parámetros críticos obtenidos en el análisis previo (valores medios de umbrales y 8 *SAQs* por grupo). En el caso de las simulaciones con *VOQnet* se asume, lógicamente, que en cada entrada o salida la memoria se divide en tantas colas como terminales en la red (64 colas, pues, como hemos mencionado sólo hemos considerado redes de 64 terminales), mientras que en el caso de *VOQsw*, la memoria se divide en tantas colas como salidas tengan los conmutadores (8 colas, por tanto). Para las simulaciones con *Virtual Channels* hemos asumido el empleo de 4 canales virtuales (por tanto la memoria de cada entrada o salida se divide en 4 colas), por ser un número normal en los dispositivos comerciales de aquellas tecnologías que los permiten[10]. Como ya se ha comentado, la política de almacenamiento en los distintos canales virtuales es seleccionar la cola más vacía en el momento de la recepción del paquete. En las simulaciones sin ninguna técnica de control de congestión, toda la memoria de una entrada o salida corresponde a una única cola, donde se almacenan todos los paquetes.

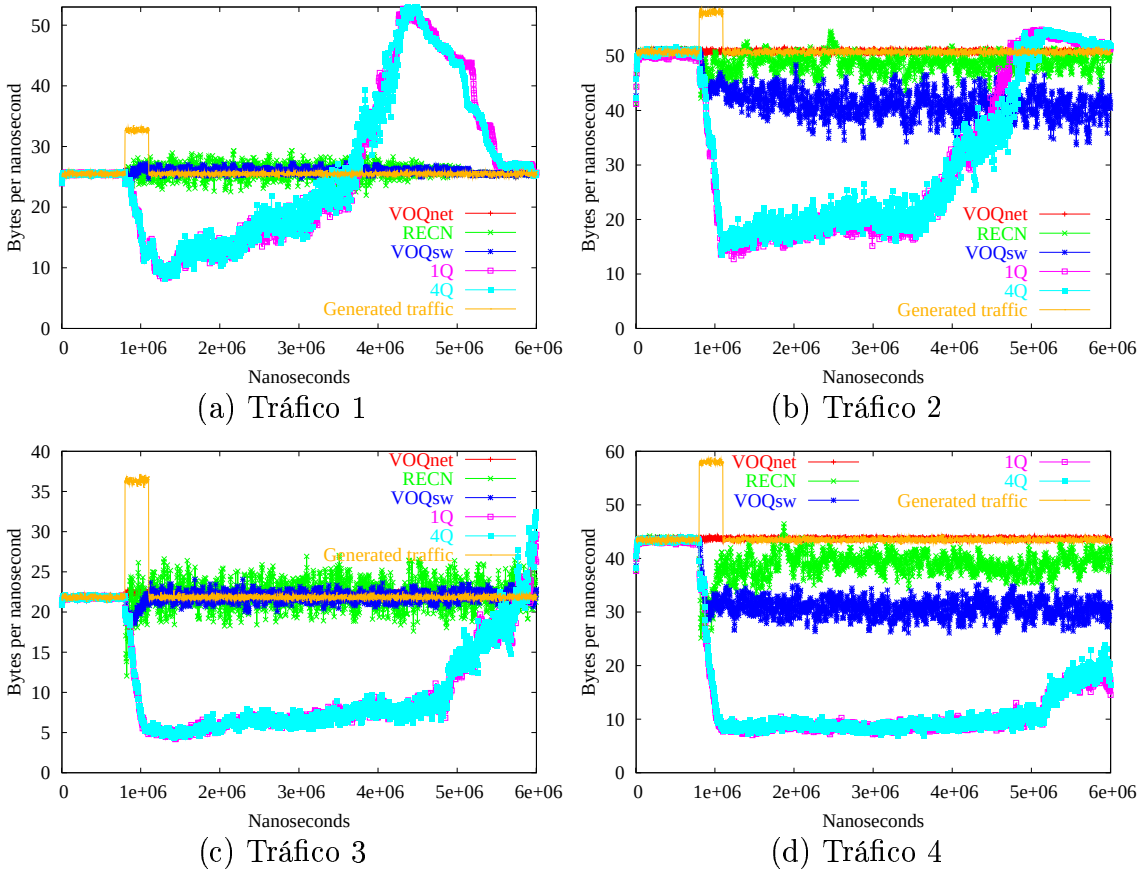
Es importante tener en cuenta que la diferente organización de la memoria de los puertos en las distintas técnicas de control de congestión tiene su reflejo también en las colas de salida de cada *Input Adapter*, pues éstas siguen el mismo esquema que exista en las colas de las salidas de los conmutadores. Así, por ejemplo, para las simulaciones donde se emplee *VOQnet*, habrá 64 colas en la salida de los *Input Adapters*.

La métrica en que hemos basado este análisis es, de nuevo, la productividad de la red, como medida de hasta qué punto las distintas técnicas son capaces de eliminar el *HOL blocking*. Naturalmente, hemos obtenido medidas de la productividad en la red para cada combinación de los distintos casos de tráfico, *speedups* y técnicas de control de congestión considerados. Los resultados obtenidos se exponen y comentan en la siguiente sección.

4.4.4.2. Resultados

La figura 4.23 muestra los resultados de productividad en función del tiempo para todos los casos de tráfico sintético considerados y para el caso de que los conmutadores de la red tengan un valor de *speedup* de 1,5. Cada gráfica individual corresponde a un único caso de tráfico, y contiene varias series, cada una con los resultados de productividad de una técnica de control de congestión concreta para dicho caso de tráfico. Nótese que la serie correspondiente al uso de 4 canales virtuales aparece en las leyendas de las gráficas como **4Q**, mientras que la correspondiente a no usar ninguna técnica de control de congestión aparece como **1Q**. Además, y como en el análisis anterior, en

[10] Aunque algunos estándares permiten un mayor número de canales virtuales, la tendencia a la hora de fabricar dispositivos reales es usar un número mucho menor del máximo permitido. Esta tendencia es debida al incremento en el coste que supone la implementación de una gran cantidad de canales virtuales, prefiriendo los fabricantes incrementar en su lugar el número de puertos de los conmutadores [MAG03].



(a) Tráfico 1 (b) Tráfico 2

(c) Tráfico 3 (d) Tráfico 4

Figura 4.23: Productividad de la red en función del tiempo para los casos de tráfico sintético 1 a 4 y para las distintas técnicas de control de congestión consideradas. Conmutadores con *speedup*=1,5.

cada gráfica individual se incluye una serie que representa el tráfico generado en la red a lo largo de la simulación.

Como puede verse en la figura 4.23, los resultados de productividad obtenidos con *VOQnet* son los únicos que, en todos los casos, siguen fielmente el ritmo del tráfico inyectado, con la excepción, naturalmente, del intervalo de inyección de tráfico congestionado. Es decir, los resultados de *VOQnet* marcan el máximo de productividad que puede alcanzarse. Teniendo esto en cuenta, puede comprobarse que, del resto de técnicas, *RECN* es la que consigue los resultados que más se aproximan a los obtenidos con *VOQnet*, igualándolos en varios casos y con una diferencia máxima de aproximadamente un 10 % para el caso peor (caso de tráfico 4). Estos resultados de *RECN* son positivos teniendo en cuenta que *VOQnet* es un técnica prácticamente ideal y muy difícilmente implementable (incluso en esta red, cuyo tamaño puede considerarse medio, *VOQnet* necesita 64 colas por puerto de entrada o salida). De otro modo, *RECN* consigue los mejores resultados de aquellas técnicas cuya implementación es asequible.

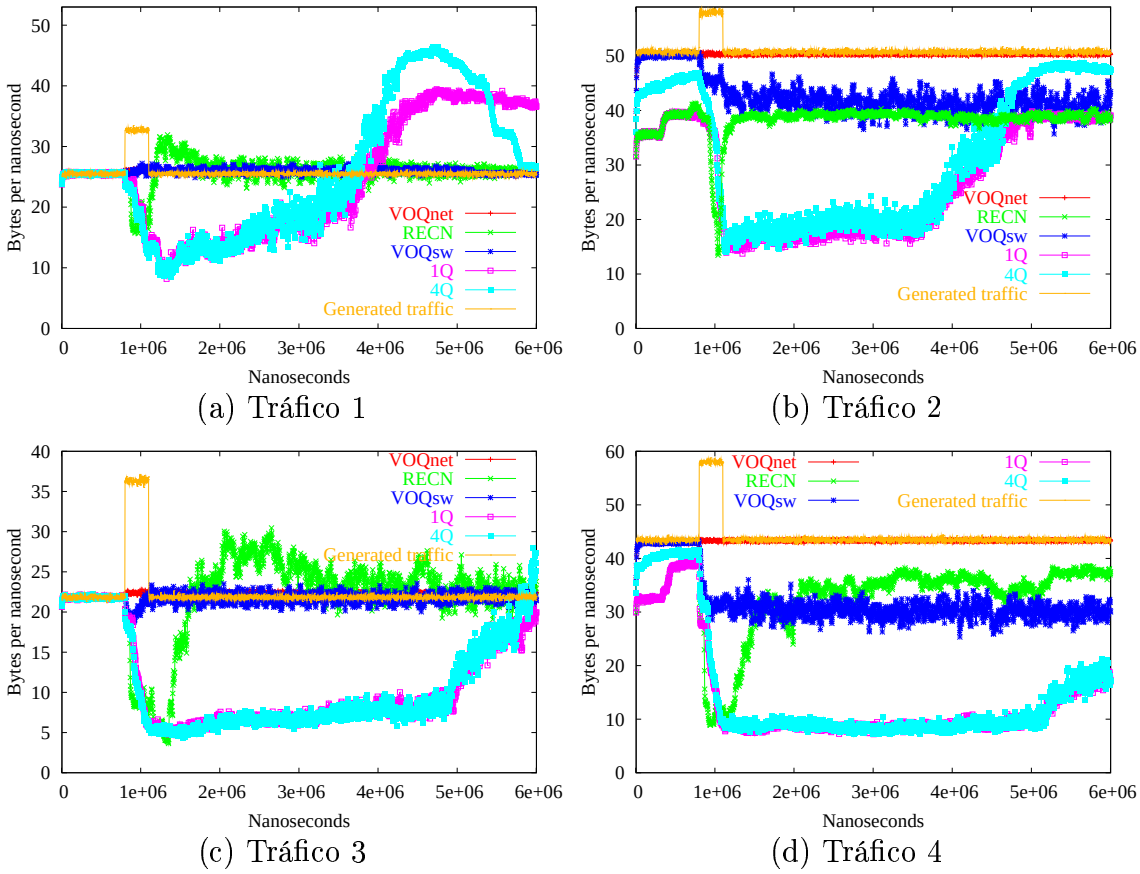
Concretamente, *RECN* ofrece en todos los casos prestaciones muchísimo mejores que las obtenidas con el uso de canales virtuales y que (como era previsible) las obtenidas sin usar técnicas de control de congestión. En este sentido, es interesante resaltar que, sin ninguna técnica de control de congestión, la productividad de la red puede llegar a caer de forma espectacular (hasta un 80 % para los casos de tráfico 2 y 4) cuando comienza a generarse tráfico congestionado, lo cual da una idea de hasta qué punto el *HOL blocking* es un fenómeno indeseable. También puede observarse en las gráficas que, en estos casos, existe un aumento repentino de la productividad mucho después de haber descendido. Esto es debido a que los paquetes de tráfico uniforme que sufren *HOL blocking* permanecen almacenados en las colas de los puertos (que tienen una gran capacidad al existir sólo una única cola y 4 colas por puerto, respectivamente) mientras que el tráfico congestionado entorpece su avance. Pero una vez que el tráfico congestionado desaparece, todos los paquetes de tráfico uniforme almacenados pueden salir de la red rápidamente (puesto que ya están generados e inyectados en la red), lo cual origina esos "picos" de productividad.

Respecto a los resultados obtenidos para *VOQsw*, hay que decir que para los casos de tráfico menos intensos (casos 1 y 3), se aproximan a los obtenidos tanto con *RECN* como con *VOQnet*. Ahora bien, para los casos de tráfico 2 y 4, la productividad obtenida para *VOQnet* desciende un 20 % y un 30 %, respectivamente, cuando comienza a generarse tráfico congestionado, y no se recupera durante toda la simulación. En ambos casos de tráfico, *RECN* soporta mucho mejor la existencia de tráfico congestionado, por lo que podemos concluir que, al menos en los escenarios considerados hasta el momento, *RECN* elimina el *HOL blocking* producido por la congestión de forma mucho más eficaz que *VOQsw*.

Evidentemente, aunque estos resultados son prometedores, es conveniente comprobar la validez de *RECN* en otros escenarios. Para ello consideraremos ahora el caso de que los conmutadores de la red tengan un valor de *speedup* de 1. La figura 4.24 muestra los resultados de productividad en función del tiempo para este valor de *speedup* y para todos los casos de tráfico sintético, siguiendo un esquema análogo al de la figura 4.23.

A la vista de estos resultados, es evidente que la eficacia de *RECN* se ve afectada negativamente por esta disminución del *speedup* de los conmutadores. Aunque para algún caso de tráfico (tráfico 1) el mecanismo consigue prestaciones aceptables, en los casos de tráfico intenso (sobre todo los casos 2 y 4) las prestaciones obtenidas con *RECN* se alejan demasiado de las obtenidas por *VOQnet*. Puede observarse en estos casos que durante el intervalo de generación de tráfico congestionado, la productividad obtenida con *RECN* llega a caer hasta un 80 %, aproximadamente, y que, aunque se recupera tras cierto tiempo, sigue siendo muy inferior al valor ideal (un 20 % menor, aproximadamente).

Más aún, aunque *RECN* consigue en todos los casos mejores prestaciones que las obtenidas para *Virtual Channels* y que las obtenidas sin usar control de congestión, no



(a) Tráfico 1 (b) Tráfico 2

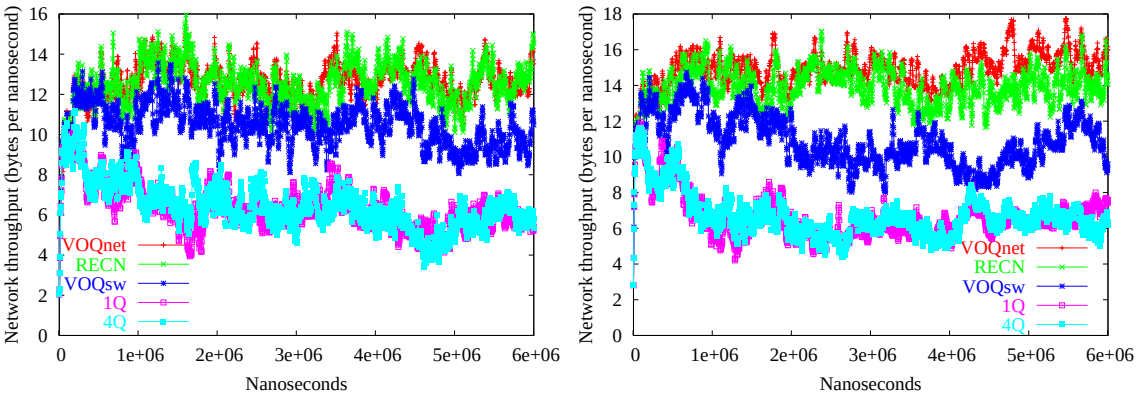
(c) Tráfico 3 (d) Tráfico 4

Figura 4.24: Productividad de la red en función del tiempo para los casos de tráfico sintético 1 a 4 y para las distintas técnicas de control de congestión consideradas. Conmutadores con *speedup*=1.

consigue superar a *VOQsw* casi en ningún momento. Esta técnica incluso presenta una mayor eficacia que *RECN* durante el intervalo de generación de tráfico congestionado (allí donde los resultados de *RECN* sufren el mayor bache).

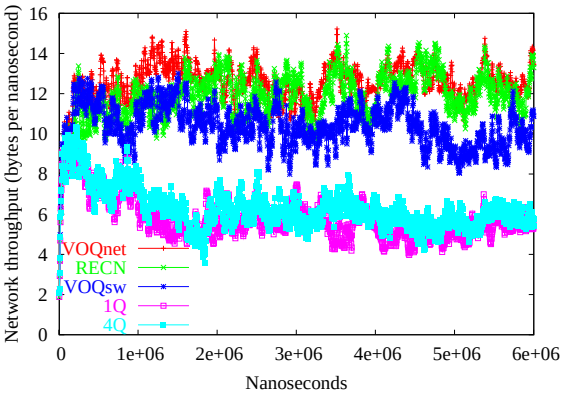
Si se comparan las figuras 4.23 y 4.24, puede comprobarse que, a excepción de *RECN*, ninguna de las técnicas parece verse excesivamente afectada por el cambio en el valor del *speedup*. Esto resulta especialmente interesante, pues indica que este cambio no introduce necesariamente situaciones de congestión mucho más complicadas de resolver (como es lógico, por otra parte). Esto implica, por tanto, que de algún modo el planteamiento de *RECN* respecto al tratamiento de la congestión hace que la eficacia de este mecanismo dependa de la configuración del conmutador (o al menos del valor de *speedup* del mismo).

Para confirmar esta idea, seguiremos adelante con el análisis mostrando en esta ocasión los resultados obtenidos usando ficheros de trazas como carga de tráfico. La figura 4.25 muestra los resultados obtenidos para las trazas (con distintos factores de

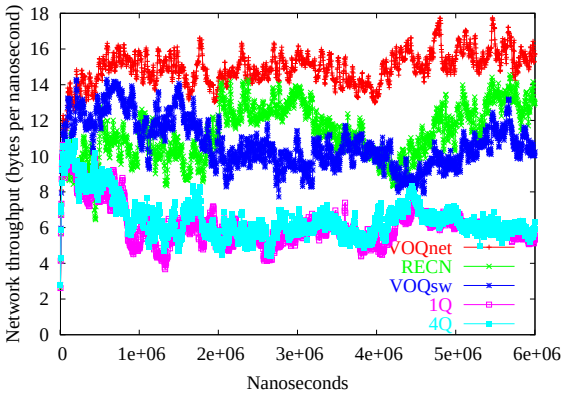


(a) FC 20, *speedup*=1,5

(b) FC 40, *speedup*=1,5



(c) FC 20, *speedup*=1



(d) FC 40, *speedup*=1

Figura 4.25: Productividad de la red en función del tiempo para las trazas con factores de compresión 20 (a y c) y 40 (b y d) y para las distintas técnicas de control de congestión consideradas. Conmutadores con *speedup*=1, 5 (a y b) y *speedup*=1 (c y d).

compresión) y para distintos valores de *speedup* de los conmutadores. Cada gráfica individual muestra por tanto resultados de cada una de las técnicas de control de congestión para un valor concreto de *speedup* del conmutador y para un factor de compresión concreto de las trazas.

Como puede verse en la figura, los resultados para trazas siguen la misma tendencia que los mostrados en figuras anteriores. Como era previsible, los mejores resultados corresponden en todos los casos a *VOQnet*, que marca el nivel ideal de productividad. Los resultados obtenidos con *RECN* para conmutadores con valor de *speedup* de 1,5 (4.25.a y 4.25.b) se aproximan bastante al nivel ideal, igualándolo en muchas ocasiones, y superando así en estos casos a los resultados obtenidos con el resto de técnicas (que, por otra parte, presentan un comportamiento muy similar al observado en los resultados para tráfico sintético). Los mismo sucede (4.25.c) para el caso de conmutadores con valor de *speedup* de 1 si el tráfico no es demasiado intenso (factor de compresión 20). Pero de nuevo puede observarse (4.25.d) que cuando el tráfico es más intenso (factor

de compresión 40) y los conmutadores no presentan aceleración interna, la eficacia de *RECN* se resiente, y los resultados obtenidos para este mecanismo se alejan demasiado del nivel ideal, e incluso en algunos instantes son superados por los resultados de *VOQsw*. Nótese además que, de nuevo, sólo *RECN* parece verse afectado por el cambio del valor de *speedup*.

En definitiva, los resultados obtenidos para trazas confirman las conclusiones extraídas a partir de la observación de los resultados obtenidos para tráfico sintético: *RECN* presenta un excelente comportamiento para redes en las que los conmutadores presentan aceleración interna (igualando la eficacia de *VOQnet* y superando al resto de técnicas), pero no sucede lo mismo en redes con conmutadores sin aceleración interna, donde la eficacia de *RECN* se degrada hasta niveles indeseables.

Teniendo en cuenta estas conclusiones, es evidente que *RECN* debería modificarse de algún modo para mejorar las deficiencias detectadas. Ahora bien, las conclusiones anteriores son válidas para redes de tamaño medio, como las consideradas en este análisis, y, en principio, no podemos considerarlas directamente válidas para redes de mayor tamaño. Además, tampoco hemos analizado la cantidad de *SAQs* por puerto realmente empleadas por *RECN* (sólo sabemos la máxima cantidad posible) en los distintos casos de tráfico considerados, lo cual nos impide valorar la escalabilidad del mecanismo. Para cerrar estas cuestiones pendientes, continuaremos con el previsto análisis de escalabilidad.

4.4.5. Análisis de escalabilidad

Con este tipo de análisis se pretende comprobar si las prestaciones ofrecidas por un mecanismo se mantienen sea cual sea el tamaño de la red sin que para ello sea necesario emplear una mayor cantidad de recursos. En el caso de *RECN*, su escalabilidad viene determinada por la necesidad o no de emplear más *SAQs* por puerto para mantener la productividad (o lo que es lo mismo, para seguir eliminando el *HOL blocking* eficientemente) cuando se aumenta el tamaño de la red.

En consecuencia, este análisis enlaza directamente con el anterior, pues los resultados obtenidos anteriormente para redes de tamaño medio deben compararse con los obtenidos para redes mayores. Además, también debe compararse el consumo de *SAQs* para ambos tamaños, en distintos casos de tráfico.

Por otra parte, es evidente que el presente análisis puede realizarse a partir de resultados obtenidos exclusivamente para *RECN*, no siendo realmente necesario establecer comparaciones con los resultados obtenidos para otras técnicas. A pesar de ello, se ha considerado oportuno comparar los resultados obtenidos para *RECN* con los obtenidos para *VOQsw*, pero sólo para tener una referencia de la eficacia de una técnica que, como *RECN*, requiere una cantidad de recursos que no depende del tamaño de la red.

		Tráfico uniforme			Tráfico árbol de congestión				
Caso de tráfico	Tamaño de red (BMIN)	Nº. de fuentes	Destino	Tasa de generación	Nº. de fuentes	Destino	Tasa de generación	Instante inicio	Instante fin
512-1	512×512	75 %	aleatorio	100 %	25 %	Terminal 32	100 %	800 μs	1100 μs
512-2	512×512	75 %	aleatorio	100 %	6.25 %	Terminal 32	100 %	800 μs	1100 μs
					6.25 %	Terminal 201	100 %	800 μs	1100 μs
					6.25 %	Terminal 428	100 %	800 μs	1100 μs
					6.25 %	Terminal 500	100 %	800 μs	1100 μs
2048-1	2048×2048	75 %	aleatorio	100 %	25 %	Terminal 32	100 %	800 μs	1100 μs
2048-2	2048×2048	75 %	aleatorio	100 %	6.25 %	Terminal 32	100 %	800 μs	1100 μs
					6.25 %	Terminal 515	100 %	800 μs	1100 μs
					6.25 %	Terminal 1540	100 %	800 μs	1100 μs
					6.25 %	Terminal 2000	100 %	800 μs	1100 μs

Tabla 4.3: Tráficos empleados en el análisis de escalabilidad para redes de 512 y 2.048 terminales.

4.4.5.1. Configuración de las pruebas

En las pruebas correspondientes al presente análisis se han empleado, como no podía ser de otro modo, redes de distintos tamaños. En concreto, y teniendo en cuenta que se dispone ya de resultados para *BMINs* de 64 terminales (las únicas consideradas en el análisis anterior), se han realizado pruebas adicionales para *BMINs* de 512 y de 2.048 terminales. Para no desviar el análisis de su enfoque principal en el tamaño de la red, y para que la comparación de prestaciones se realice en condiciones lo más similares posible, no se ha considerado oportuno variar en las nuevas pruebas las características de los conmutadores respecto a las asumidas en las pruebas anteriores. Por tanto, se continúa asumiendo que la red se construye mediante conmutadores de 8 puertos bidireccionales, lo cual supone que la *BMIN* de 512 terminales tenga 5 etapas, con 128 conmutadores por etapa, y que la *BMIN* de 2.048 terminales tenga 6 etapas, con 512 conmutadores por etapa. Además, se sigue asumiendo que la memoria de los puertos de entrada o salida de los conmutadores es de 32 KB. Respecto al *speedup*, al igual que en las pruebas del anterior análisis, las nuevas pruebas se han realizado considerando bien conmutadores con *speedup* de valor 1,5, bien conmutadores con *speedup* de valor 1.

Respecto al tráfico empleado en este análisis, desgraciadamente, para las redes de 512 y 2.048 terminales no es posible realizar pruebas con trazas como carga de tráfico, al no disponer de ficheros de trazas originados en redes de este tamaño. Por el contrario, estos mayores tamaños de red permiten una mayor variedad en los patrones de tráfico sintético, al menos en cuanto al modo en que éstos modelan árboles de congestión. Así, si en una red de 64 terminales es poco realista modelar varios árboles de congestión intensos y simultáneos (ya que en ese caso prácticamente no quedarían fuentes de tráfico no congestionado), en redes de mayor tamaño este problema no existe. Teniendo esto en cuenta, los patrones de tráfico empleados en la realización de las nuevas pruebas de este análisis han sido los que aparecen en la tabla 4.3.

Puede apreciarse que tanto el tráfico 512-1 como el tráfico 2048-1 siguen un esquema análogo al tráfico 4 empleado en las pruebas anteriores con redes de 64 terminales (75 % de fuentes inyectando tráfico aleatorio durante toda la simulación, al 100 % de la capacidad del enlace, y 25 % de fuentes inyectando tráfico congestionado hacia un único destino durante 300 μs). Los tráficos 512-2 y 2048-2, en cambio, siguen un esquema donde existen cuatro destinos para el tráfico congestionado, lo cual debería originar al menos 4 árboles de congestión. En estos casos, una única fuente de tráfico congestionado inyecta siempre tráfico hacia un único destino congestionado de los cuatro posibles. Nótese que el porcentaje total de fuentes inyectando tráfico congestionado es el mismo en todos los casos de tráfico (25 %).

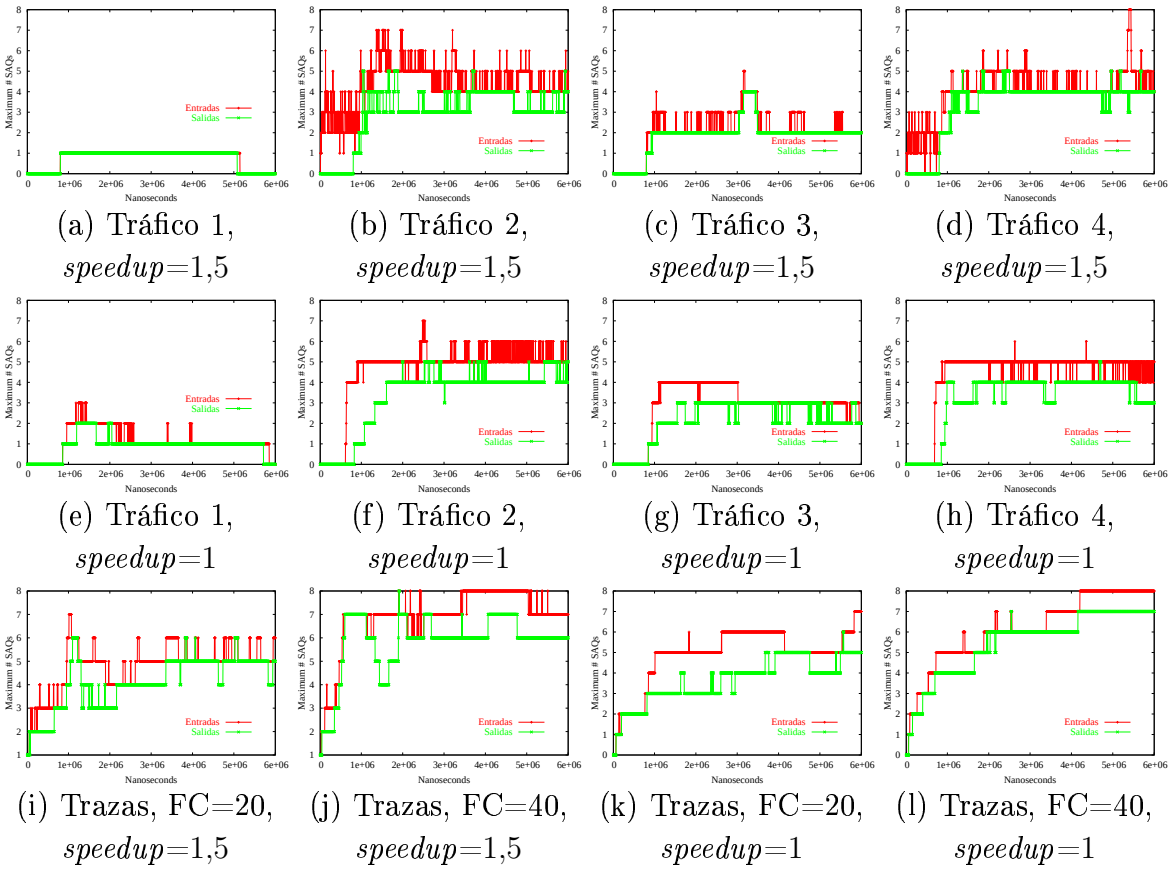
Respecto a la configuración de los mecanismos de eliminación del *HOL blocking*, se ha empleado la misma que en pruebas anteriores. Es decir, para *RECN*, hemos usado valores medios de umbrales y 8 *SAQs* por grupo. Nótese que es el mismo número de *SAQs* empleado en las pruebas para redes de 64 terminales cuyos resultados se han presentado anteriormente. Esta igualdad es imprescindible puesto que se pretende comparar las prestaciones obtenidas por *RECN* en redes de distinto tamaño cuando se emplea el mismo número de recursos. En el caso de *VOQsw*, la memoria de cada puerto de entrada o salida se divide en 8 colas. Como en casos anteriores, las colas de salida de cada *Input Adapter* siguen el mismo esquema que siguen las colas de las salidas de los conmutadores.

En cuanto a las métricas consideradas, y tratándose de un análisis sobre la escalabilidad de *RECN*, éstas no pueden ser otras que la productividad y el número de *SAQs* empleadas realmente en cada puerto de entrada o salida por el mecanismo. Obviamente, valoraremos la escalabilidad de *RECN* en la medida en que el mecanismo sea capaz de mantener el nivel de productividad para los distintos tamaños de red, usando un número similar de *SAQs*.

4.4.5.2. Resultados

Antes de presentar los resultados de las pruebas realizadas específicamente para este análisis con redes de gran tamaño, es preciso considerar el número de *SAQs* empleadas realmente por *RECN* en los casos considerados anteriormente para redes menores. Para ello, la figura 4.26 muestra distintas gráficas que corresponden a cada uno de los casos de tráfico (sintético y trazas) y de valor de *speedup* considerados en el análisis anterior para redes de 64 terminales. Como puede apreciarse, en cada gráfica existen dos series que representan respectivamente el número máximo de *SAQs* activas en una entrada y el número máximo de *SAQs* activas en una salida de la red, en ambos casos a lo largo del tiempo de la simulación.

Como puede comprobarse, casi en ningún caso llegan a utilizarse todas las *SAQs* disponibles en las entradas o salidas. Esto implica que el número máximo de *SAQs*



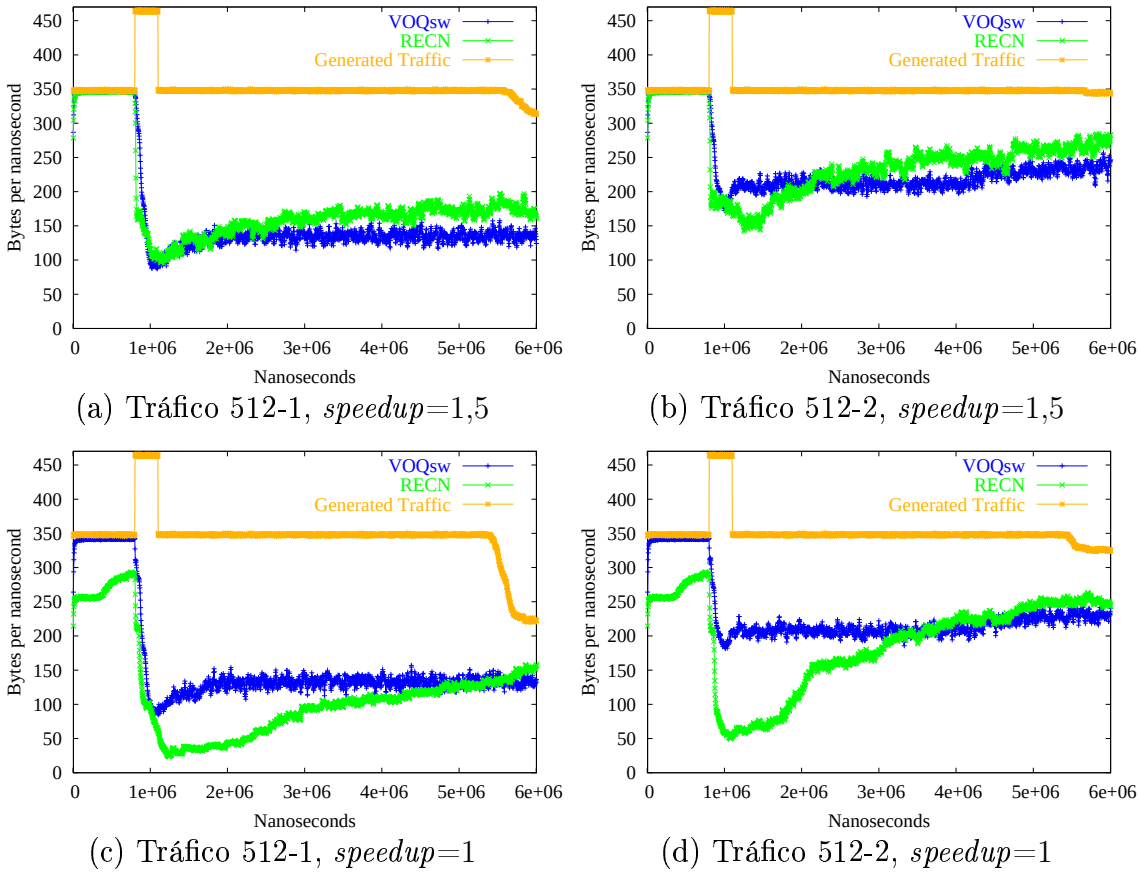
(a) Tráfico 1, *speedup*=1,5 (b) Tráfico 2, *speedup*=1,5 (c) Tráfico 3, *speedup*=1,5 (d) Tráfico 4, *speedup*=1,5

(e) Tráfico 1, *speedup*=1 (f) Tráfico 2, *speedup*=1 (g) Tráfico 3, *speedup*=1 (h) Tráfico 4, *speedup*=1

(i) Trazas, FC=20, *speedup*=1,5 (j) Trazas, FC=40, *speedup*=1,5 (k) Trazas, FC=20, *speedup*=1 (l) Trazas, FC=40, *speedup*=1

Figura 4.26: Número máximo de *SAQs* activas en entradas y salidas en función del tiempo para los casos de tráfico sintético 1 a 4 (a-d para conmutadores con *speedup*=1,5, e-h para conmutadores con *speedup*=1) y para trazas con distinto factor de compresión (i-j para conmutadores con *speedup*=1,5, k-l para conmutadores con *speedup*=1).

establecido (8) es más que suficiente para obtener las prestaciones que *RECN* ofrece en estos casos. Nótese que esto implica que la razón de que, en algunas situaciones, *RECN* no elimine completamente el *HOL blocking*, no es la falta de recursos. Recordemos que este problema aparece si los conmutadores tienen *speedup*=1, para los casos de tráfico sintético 2 y 4 y para el caso de trazas con factor de compresión 40. El número de *SAQs* empleado en estas situaciones aparece respectivamente en las gráficas 4.26.f, 4.26.h y 4.26.l. Si comparamos con el resultado obtenido para estos mismos casos de tráfico, pero con conmutadores con *speedup*=1,5 (gráficas 4.26.b, 4.26.d y 4.26.j, respectivamente), tendremos que en estos últimos casos, el número de *SAQs* empleadas es ligeramente mayor. Puesto que en el anterior análisis se pudo comprobar que *RECN* funcionaba perfectamente en estos últimos casos, cabe concluir que el mal funcionamiento de *RECN* detectado para conmutadores con *speedup*=1 no sólo no se debe a la falta de *SAQs*, sino que debe ser más bien una consecuencia de no utilizar todas las *SAQs* que debieran activarse para eliminar el *HOL blocking*. Esta observación, como veremos, es fundamental para solucionar los problemas detectados en el mecanismo.



(a) Tráfico 512-1, *speedup*=1,5

(b) Tráfico 512-2, *speedup*=1,5

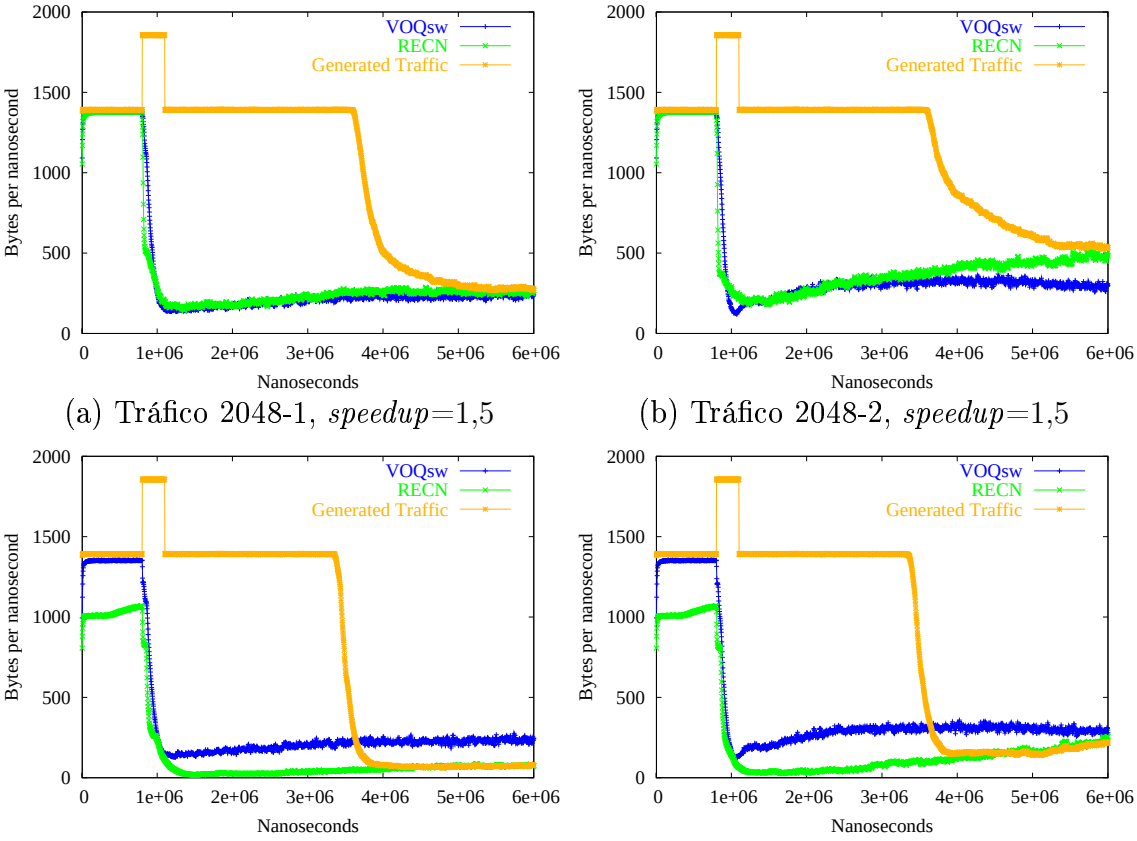
(c) Tráfico 512-1, *speedup*=1

(d) Tráfico 512-2, *speedup*=1

Figura 4.27: Productividad de la red en función del tiempo para los casos de tráfico sintético 512-1 y 512-2, y para conmutadores con *speedup*=1,5 y *speedup*=1.

Las siguientes figuras muestran ya resultados obtenidos mediante simulaciones realizadas con redes de mayor tamaño. La figura 4.27 muestra en concreto los resultados para la *BMIN* de 512 terminales. Cada gráfica individual muestra, para un caso concreto de tráfico y un valor concreto de *speedup*, la productividad de la red a lo largo del tiempo de simulación, con una serie que corresponde a la productividad obtenida usando *RECN* como técnica de control de congestión y otra serie que corresponde a la productividad obtenida usando *VOQsw*. Además, y como referencia, cada figura también contiene una tercera serie que representa el tráfico generado en la red a lo largo del tiempo de simulación.

Como puede observarse, los resultados no son en absoluto los deseables, obteniendo *RECN* unas prestaciones muy pobres. Puede apreciarse que la productividad sufre un descenso drástico (de hasta un 90 %) en todos los casos cuando comienza a inyectarse tráfico congestionado. Esto implica, obviamente que en estos casos *RECN* elimina el *HOL blocking* de forma muy poco eficaz. También puede observarse que estas prestaciones son similares a las obtenidas con *VOQsw*, por lo que *RECN* ni siquiera parece



(a) Tráfico 2048-1, *speedup*=1,5

(b) Tráfico 2048-2, *speedup*=1,5

(c) Tráfico 2048-1, *speedup*=1

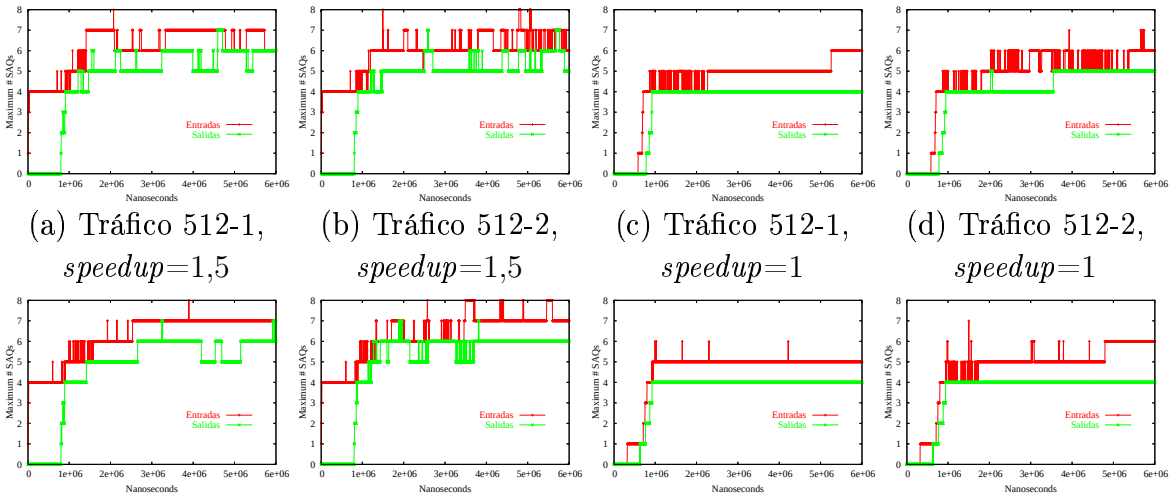
(d) Tráfico 2048-2, *speedup*=1

Figura 4.28: Productividad de la red en función del tiempo para los casos de tráfico sintético 2048-1 y 2048-2, y para conmutadores con *speedup*=1,5 y *speedup*=1.

mejorar esta propuesta. Nótese que la congestión llega incluso al punto de afectar (mediante el control de flujo) al tráfico generado desde los terminales, que sufre un ligero descenso hacia el final de la simulación. Por otra parte, no parece tener gran influencia el hecho de que existan uno o varios árboles de congestión simultáneos, pero sí el hecho de que los conmutadores tengan distinto valor de *speedup*, ya que los resultados que aparecen en las gráficas 4.27.a y 4.27.b (obtenidos para conmutadores con *speedup*=1,5), aun siendo malos, son mejores que los que pueden verse en las gráficas 4.27.c y 4.27.d (obtenidos para conmutadores con *speedup*=1).

La figura 4.28 presenta una estructura análoga a la de la figura anterior, pero mostrando en esta ocasión resultados de productividad a lo largo del tiempo para la *BMIN* de 2.048 terminales. Puede apreciarse que los resultados son similares a los mostrados en la figura anterior[11]. Se observa incluso una caída mayor de la productividad, lo que

[11] Cabría pensar que todos estos malos resultados pudieran deberse a que los umbrales críticos se han ajustado para redes de 64 terminales, y no para redes mayores. Sin embargo, hemos comprobado que, utilizando ternas de valores de umbrales altos o bajos, los resultados para redes grandes son aún peores que los mostrados.



(a) Tráfico 512-1, *speedup*=1,5 (b) Tráfico 512-2, *speedup*=1,5 (c) Tráfico 512-1, *speedup*=1 (d) Tráfico 512-2, *speedup*=1

(e) Tráfico 2048-1, *speedup*=1,5 (f) Tráfico 2048-2, *speedup*=1,5 (g) Tráfico 2048-1, *speedup*=1 (h) Tráfico 2048-2, *speedup*=1

Figura 4.29: Número máximo de *SAQs* activas en entradas y salidas en función del tiempo para los casos de tráfico sintético 512-1 y 512-2 (a y b para conmutadores con *speedup*=1,5, c y d para conmutadores con *speedup*=1) y para los casos 2048-1 y 2048-2 (e y f para conmutadores con *speedup*=1,5, g y h para conmutadores con *speedup*=1).

indica una presencia más significativa del *HOL blocking* en la red. También puede apreciarse que, aun siendo malos todos los resultados de *RECN*, los peores corresponden de nuevo a aquellos casos en los que los conmutadores no tienen aceleración interna (*speedup*=1).

Considerando el análisis que estamos realizando, la pregunta que debe plantearse inmediatamente, a la vista de la pobreza de los resultados de *RECN* para redes de gran tamaño, es si estas pobres prestaciones se deben a la falta de recursos. En definitiva, cabría plantearse si 8 *SAQs* en cada puerto de entrada o salida no son suficientes en redes grandes para eliminar el *HOL blocking*. De cara a responder a esta cuestión, la figura 4.29 muestra, siguiendo un esquema similar al de la figura 4.26, el número máximo de *SAQs* activas en entradas y salidas para los casos de tráfico y de valor de *speedup* que se han considerado para redes de 512 y 2.048 terminales.

Como puede apreciarse, casi en ningún caso se llega a activar el número máximo de *SAQs* disponibles en entradas o salidas. Por tanto, la escasa eficacia mostrada por *RECN* en estos casos, de nuevo no puede deberse a la falta de recursos. Nótese también que, como en casos anteriores, en las situaciones donde *RECN* presenta un peor comportamiento, el consumo de *SAQs* es menor.

A la vista de todos estos resultados, no podemos extraer conclusiones precisas sobre la escalabilidad de *RECN*, ya que realmente los resultados no son los deseables sea cual sea el tamaño de la red. En cualquier caso, sí puede decirse que los fallos detectados

en el mecanismo no se deben a que los recursos destinados a eliminar el *HOL blocking* sean escasos, sino más bien a que los recursos disponibles no se aprovechan plenamente como deberían.

4.5. Conclusiones

A la vista de los resultados de productividad presentados en los distintos análisis en los que hemos basado la evaluación de *RECN*, es evidente que el mecanismo debe ser mejorado. Aunque en algún escenario (redes pequeñas con conmutadores con aceleración interna) *RECN* se comporta razonablemente bien, eliminando casi totalmente el *HOL blocking* producido por los paquetes congestionados, su eficiencia baja en redes grandes o con conmutadores sin aceleración interna.

Por otra parte es fundamental destacar que los resultados del número de *SAQs* activas en los puertos parecen indicar que esta pérdida de eficacia de *RECN* se debe a un mal aprovechamiento de los recursos disponibles para eliminar el *HOL blocking* (las *SAQs* en cada entrada o salida), que en ningún caso se agotan, y que son consumidos en menor medida en aquellas situaciones en las que *RECN* presenta un peor comportamiento. En definitiva, por alguna razón *RECN* no activa como debería las *SAQs* en ciertas ocasiones en que esto es necesario.

De esto se deduce que el planteamiento inicial de *RECN*, expuesto en el presente capítulo, presenta algún punto débil que impide que la técnica sea realmente eficiente. Afortunadamente, el problema fue detectado tras un análisis exhaustivo de numerosas pruebas de simulación. Dicho problema se explica en el siguiente capítulo, junto con las mejoras que deben introducirse en *RECN* para solucionarlo.

Por tanto, téngase en cuenta que la versión de *RECN* presentada con detalle en este capítulo corresponde al planteamiento inicial de nuestra técnica de control de congestión, y que la versión final del mecanismo (que sí es realmente eficiente y escalable) es fruto de mejorar considerablemente esta versión inicial mediante los cambios que se explican en el siguiente capítulo.

Capítulo 5

Mejoras de la propuesta inicial

Como se ha podido comprobar en el capítulo precedente, la propuesta inicial de *RECN* necesita mejorarse para alcanzar el objetivo de ser una técnica realmente eficaz y escalable para la eliminación del *HOL blocking*. En el presente capítulo se detallan las distintas mejoras introducidas en *RECN* de cara a que dicho objetivo se cumpla satisfactoriamente.

Aunque los nuevos mecanismos expuestos en el presente capítulo son ciertamente cambios o añadidos a la propuesta inicial, conviene destacar que la propuesta de estas técnicas, fruto de los análisis que nos condujeron a determinar los defectos del planteamiento original, constituye una fase fundamental de nuestra investigación, pues ésta no sólo nos ha permitido solucionar los problemas de *RECN* y obtener una versión final del mismo plenamente eficiente, sino que también nos ha llevado a una mejor comprensión de la dinámica de la congestión en redes de interconexión, lejos de las ideas clásicas sobre este fenómeno.

Por tanto, este capítulo comienza explicando los defectos detectados en la propuesta inicial, que básicamente se deben a una concepción errónea de los procesos de formación y desaparición de los árboles de congestión, cuya variada y compleja evolución se detalla convenientemente. Seguidamente se analiza el impacto negativo de algunos tipos de evolución de árboles de congestión en la propuesta inicial de *RECN*, lo cual, por una parte, justifica totalmente la introducción de mejoras y, por otra, determina las necesidades y carencias que éstas deben solventar. Lógicamente, a continuación se describen exhaustivamente cada una de las mejoras introducidas.

Como es lógico, en el presente capítulo también se presentan resultados que permiten evaluar las prestaciones ofrecidas por la versión de *RECN* mejorada, de forma similar a como lo fuera la versión inicial en el capítulo precedente. Por último, el capítulo muestra las conclusiones que pueden extraerse de dichos resultados y que, como se verá, hacen evidentes las grandes ventajas del uso de la versión final de *RECN*.

5.1. Justificación de las mejoras

Los pobres resultados obtenidos en algunas circunstancias por la propuesta inicial de *RECN*, que pueden observarse en el capítulo anterior, justifican ya de por sí el intentar mejorar dicha propuesta. Ahora bien, es evidente que las mejoras deben encaminarse a subsanar los defectos que pudiera tener el mecanismo en sus distintas facetas, y para ello es imprescindible conocer precisamente y con exactitud en qué aspectos la propuesta inicial es errónea. Sólo desde este conocimiento es posible proponer cambios en *RECN* que solucionarían los mencionados problemas, y cuya inclusión estaría de este modo realmente justificada.

Siguiendo este planteamiento, el primer paso para la corrección de la propuesta consistió en un análisis de aquellos casos en los que la eficacia de *RECN* disminuía, con la intención de encontrar un motivo para esta pérdida de prestaciones. Como puede verse en los resultados mostrados en el capítulo precedente, una circunstancia que disminuye en todos los casos la eficacia de *RECN* es la ausencia de aceleración interna en los conmutadores de la red (conmutadores con *speedup*=1). Además, en los resultados también se aprecia que en estos casos el número de *SAQs* activas, en general, es menor en toda la red. Ambas observaciones nos llevaron a pensar que, de algún modo, los árboles de congestión no eran correctamente detectados o notificados cuando se empleaban conmutadores sin aceleración interna, no siendo posible así la activación de las *SAQs* necesarias para eliminar el *HOL blocking* que dichos árboles pudieran producir.

Buscando las razones de esta supuestamente incorrecta detección de árboles de congestión para este tipo de conmutadores, descubrimos un defecto fundamental de la propuesta inicial de *RECN*: la detección de congestión sólo se realiza en las salidas de los conmutadores, mientras que en conmutadores sin aceleración interna, la congestión aparece en las entradas de los mismos[1]. Es decir, la propuesta inicial de *RECN* puede no detectar situaciones de congestión en su punto de origen (la raíz del árbol).

Este descubrimiento no sólo nos llevó a replantearnos la forma en que debería realizarse la detección de congestión, sino que también nos hizo sospechar que la eficacia de *RECN* dependía de la forma en que evolucionaran los árboles de congestión que pudieran aparecer en la red. Aún más allá, se consideró necesario revisar nuestra propia visión sobre el crecimiento de los árboles de congestión, la cual, en definitiva, había influenciado el diseño de *RECN* hasta entonces.

Esto nos obligó a realizar un completo estudio de las distintas formas en que los árboles de congestión pueden evolucionar, tanto durante su crecimiento como durante su desaparición. Este estudio se mostrará con todo detalle en el punto siguiente, pero

[1]Evidentemente, asumiendo que el conmutador es de tipo *IQ* ó *CIOQ*. Este fenómeno ya se apuntó en la sección 2.6.2, pero se explicará en mayor profundidad en los siguientes puntos.

conviene mencionar ya las dos principales ideas erróneas en nuestra visión sobre los árboles, que dicho análisis nos permitió detectar. Estas ideas erróneas se corresponden a una concepción "tradicional" de los árboles de congestión, y son las siguientes:

- **La congestión se origina siempre en las salidas de los conmutadores**. Ya hemos mencionado que esta afirmación es falsa al menos para conmutadores sin aceleración interna de tipo *IQ* ó *CIOQ*, pero, como veremos en el punto siguiente, tampoco es cierta para conmutadores de estos tipos con aceleración interna. En definitiva, la congestión puede originarse en entradas y/o salidas de los conmutadores mientras existan colas en estos puntos y se den ciertas condiciones (no exclusivamente referidas al *speedup*).
- **Los árboles de congestión crecen siempre desde la raíz hasta las hojas, y desaparecen en sentido contrario**. Como veremos con detalle en el punto siguiente, realmente los árboles de congestión pueden evolucionar de muy distintas formas en función de la arquitectura del conmutador y de los patrones de tráfico, por lo que esta idea es totalmente errónea.

Ambas ideas, explícita o implícitamente, están reflejadas en ciertos aspectos de la propuesta original de *RECN*, como por ejemplo en la estrategia utilizada para detectar congestión, o en las condiciones requeridas para liberar *SAQs*. Como es lógico, es precisamente el hecho de que estos aspectos de la propuesta original se basen en asunciones incorrectas lo que provoca que ciertas funciones de *RECN* no se realicen de forma adecuada en determinadas circunstancias (en concreto, cuando los árboles de congestión no se comportan de acuerdo con las ideas "clásicas" expuestas), lo que a su vez es la causa de la degradación de prestaciones del mecanismo observada en dichas circunstancias.

Naturalmente, el impacto negativo de los distintos tipos de evolución de los árboles de congestión en cada uno de los aspectos "erróneos" de la propuesta original (y en *RECN* en conjunto) se analiza convenientemente en un punto posterior, pero para ello es imprescindible mostrar antes los resultados de nuestro estudio sobre la evolución de los árboles de congestión, a lo que está dedicado el punto siguiente.

5.1.1. Evolución de los árboles de congestión

Tal y como se ya se ha indicado, la evolución de los árboles de congestión en una red de interconexión puede suceder de muy variadas formas. Es decir, el modo en que estos árboles se originan, se expanden y desaparecen no es único, sino que presenta una dinámica compleja en la que influyen varios factores.

Algunos de estos factores, como la topología de la red, tienen una influencia obvia en las posibles formas de evolución de los árboles, pero existen otros factores cuya

influencia es mucho más sutil. En concreto, tanto la variación de algunas características de la arquitectura de los conmutadores de la red como ciertas combinaciones de patrones de tráfico pueden hacer que la evolución de los árboles sea muy distinta.

En los siguientes puntos se detallan las diversas formas en las que la congestión puede aparecer, expandirse (en forma de árboles de congestión) y desaparecer (con el colapso de los árboles), y la influencia de los factores mencionados en cada uno de estos aspectos, mostrando numerosos y variados ejemplos de evolución de árboles.

5.1.1.1. Aparición de puntos de congestión

La arquitectura del conmutador es un factor que influye principalmente, y en varios sentidos, en la forma en la que la congestión se origina. Por una parte, es obvio que en el caso de conmutadores de tipo *OQ*, la congestión sólo puede aparecer en las salidas, ya que sólo existen colas en estos puntos. Análogamente, en conmutadores de tipo *IQ*, la congestión sólo puede aparecer en las entradas.

Ahora bien, en el caso de los conmutadores de tipo *CIOQ* (que en definitiva son los empleados en las más modernas propuestas), no es trivial estimar si la congestión aparecerá antes en entradas o salidas, ya que en este caso existen colas en ambos lados del conmutador. De hecho, en este tipo de conmutadores es necesario conocer más datos para poder determinar el punto de aparición de la congestión.

Por ejemplo, la figura 5.1.a muestra un ejemplo donde, en un conmutador *CIOQ*, la congestión aparece en las entradas. Esto es debido a que, como indica la figura, se trata de un conmutador sin aceleración interna (*speedup*=1), y por tanto en el caso de que dos (o más) flujos entrantes lo suficientemente intensos soliciten la misma salida, los paquetes cruzarán el *crossbar* más lentamente de lo que son recibidos, y comenzarán a acumularse en las colas situadas en las entradas correspondientes. En definitiva, la congestión aparecerá en este caso en las entradas afectadas. En el ejemplo mostrado en esta figura, se ha asumido que los dos flujos entrantes se reciben en el conmutador a la máxima velocidad permitida por el enlace (por sencillez y para remarcar la intensidad de ambos flujos) , pero merece la pena destacar que la situación sería similar incluso si los flujos se recibiesen con una tasa menor[2]; la única diferencia sería que los paquetes se acumularían en las entradas más lentamente. También hay que resaltar que los paquetes nunca se acumulan en la cola de salida (a menos que actúe el control de flujo desde el siguiente conmutador), ya que la ausencia de aceleración interna en el *crossbar* hace que siempre lleguen a la salida al mismo ritmo al que pueden ser emitidos desde ésta.

Un ejemplo distinto se muestra en la figura 5.1.b, donde la situación de partida es casi idéntica a la de la figura 5.1.a, pero en este caso se asume que el conmutador sí tiene

[2]Siempre que esta tasa sea superior al 50 % del ancho de banda del enlace: en caso contrario los paquetes cruzarían el *crossbar* a más velocidad que entran.

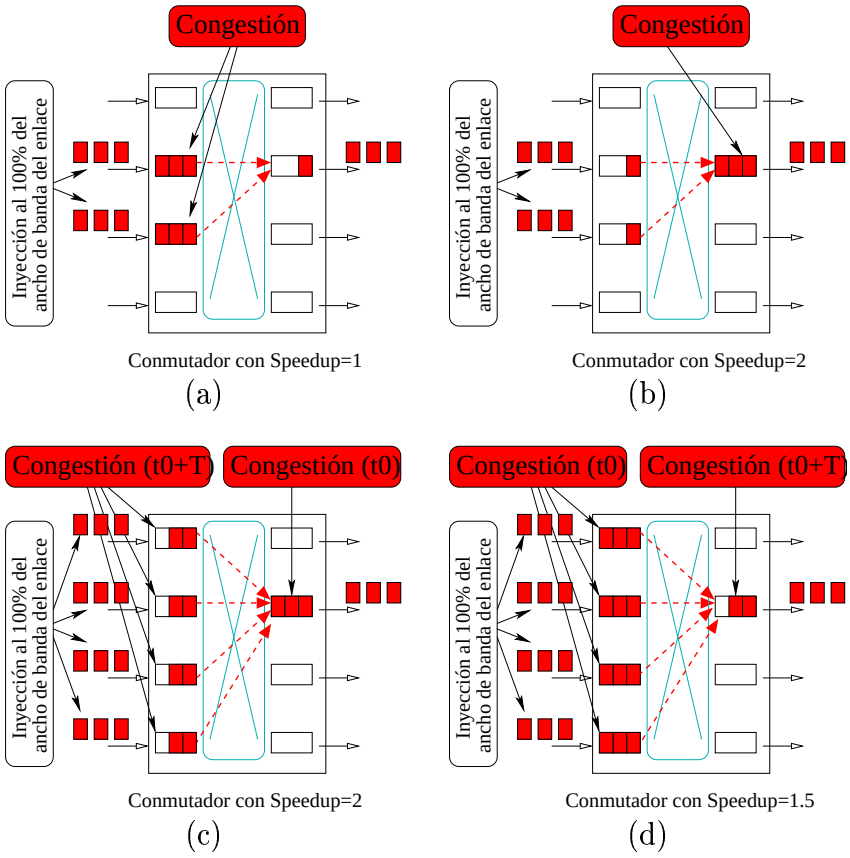


Figura 5.1: Aparición de congestión en entradas o salidas de conmutadores de tipo *CIOQ* con distintos valores de *speedup*.

aceleración interna, en concreto un valor de *speedup* de 2. Como puede comprobarse, en este caso, los paquetes pertenecientes a los dos flujos contendientes pueden cruzar hacia la salida requerida por ambos al mismo ritmo que llegan a las entradas, por lo que no se acumulan en las entradas. Por contra, la velocidad con que dichos paquetes llegan a la salida es superior a la de emisión por el enlace incluso en el mejor de los casos (sin estar el enlace bloqueado por control de flujo), por lo que se acumularán en la cola de dicha salida. Por tanto, en este caso, la congestión aparecerá primero en la salida requerida por ambos flujos. Nótese que, de persistir la inyección de los flujos implicados, los paquetes también se acabarán acumulando en las entradas, pero debido al control de flujo, que actuará cuando se sature la cola de la salida solicitada.

Ahora bien, también es posible que en un conmutador con aceleración interna los paquetes se acumulen en las entradas incluso antes de que actúe el control de flujo, como puede comprobarse en la figura 5.1.c. La situación reflejada en esta figura se diferencia de la mostrada en el ejemplo de la figura 5.1.b en que en este nuevo caso existen cuatro flujos entrantes solicitando la misma salida, en lugar de dos. Puede apreciarse que aunque el valor de *speedup* del conmutador es el mismo en ambos casos (2), la presencia de más flujos en el ejemplo de la figura 5.1.c hace que los paquetes

lleguen a las entradas a un ritmo mayor del que pueden salir de ellas, por lo que también se acumulan allí, originando congestión. Es importante destacar que en este ejemplo los paquetes se acumulan tanto en las colas de las entradas como en la cola de la salida requerida, aunque con diferente ritmo, ya que, como puede comprobarse, la acumulación será mas rápida en la salida. De hecho, puede calcularse que para conmutadores con *speedup* igual o superior a 2, la congestión aparecerá siempre antes en las salidas.

Otra situación digna de considerar es la mostrada en la figura 5.1.d. En este caso, como en el ejemplo de la figura 5.1.c, también existen cuatro flujos entrantes solicitando la misma salida del conmutador, pero ahora el conmutador presenta un valor de *speedup* de 1,5. Como en el caso anterior, los paquetes se acumularán tanto en las colas de las entradas como en la de la salida, pero puede calcularse que en esta ocasión el ritmo de acumulación de paquetes en las colas será mayor en las entradas. Dicho de otro modo, en este caso la congestión aparece primero en las entradas del conmutador. La diferencia entre la aparición inicial de congestión en uno u otro lado del conmutador, como veremos, no carece de importancia desde el punto de vista de una detección precisa de la congestión.

Evidentemente, las múltiples combinaciones posibles de distintos valores de *speedup*, y del número de flujos entrantes darían lugar a situaciones ligeramente distintas a las presentadas en los ejemplos (más aún: en cada caso, también se podría variar la tasa de llegada de los flujos entrantes para generar situaciones todavía distintas), pero en general estos casos serían mínimas variantes de los reflejados en la figura 5.1.

En definitiva, podemos concluir que en un conmutador de tipo *CIOQ*, la congestión puede aparecer en entradas o salidas, y en ambos lados con mayor o menor rapidez, dependiendo de los siguientes factores:

- **El valor de *speedup* del conmutador.** Recordemos que en conmutadores reales, este valor suele situarse en el intervalo entre 1 y 2, lo cual, como puede comprobarse en los ejemplos anteriores, es suficiente margen como para originar diferencias en cuanto al lugar y la velocidad de la aparición de la congestión.

- **El número de flujos de datos que contribuyen a formar la congestión**. Obviamente, cuantos más puertos tenga un conmutador, más posibilidades habrá de que flujos entrantes soliciten una misma salida. Pero incluso en conmutadores con un número pequeño de puertos, como los empleados en los ejemplos mostrados en la figura anterior, puede que la variación en el número de flujos entrantes sea decisiva para que la congestión aparezca en uno u otro lado, y a un ritmo u otro.

- **La velocidad relativa de llegada de los paquetes pertenecientes a los flujos que contribuyen a formar la congestión.** Téngase en cuenta que este factor no depende en absoluto de la arquitectura del conmutador, y puede ser

sumamente variable, por lo cual las posibilidades de que la congestión aparezca de distinta forma en los distintos conmutadores de una red se multiplican.

Es importante destacar que aunque la congestión pueda aparecer antes en entradas o salidas de un conmutador, su origen es siempre la contención por el uso de una salida. Por tanto, independientemente de dónde aparezca antes la congestión en el conmutador, la raíz del árbol que comienza a formarse en ese momento estará situada en la salida por cuyo acceso contienden los flujos. Nótese que esto es así incluso si la propia cola situada en la raíz no se congestiona (por ejemplo, en conmutadores sin aceleración interna).

Por supuesto, si consideramos una red compuesta por varios conmutadores de tipo *CIOQ*, la congestión puede aparecer en cada uno de ellos de distinta forma, siempre en función de los factores anteriormente mencionados. La figura 5.2 muestra algunos conmutadores[3] de una red por la que circulan flujos de paquetes que siguen ciertas rutas (indicadas por las líneas rojas en la figura). En el ejemplo se asume que todas las fuentes comienzan la inyección de paquetes simultáneamente, y que dicha inyección se produce a la máxima velocidad permitida por el enlace. También se asume que el valor de *speedup* de todos los conmutadores es de 1,5. Como puede comprobarse, la distinta interacción de los flujos en los conmutadores produce que la congestión aparezca en éstos de distinta forma. Por ejemplo, en los conmutadores 1 y 5 la congestión aparece antes en las entradas, mientras que en los conmutadores 4, 6 y 7 la congestión aparece antes en las salidas. Nótese que si se hubieran empleado conmutadores con distinto *speedup*, la situación podría ser aún más variada.

En la misma figura puede apreciarse la existencia de un árbol de congestión formado por los distintos flujos, que finalmente confluyen todos en una salida del conmutador 7, donde se sitúa la raíz del árbol. Es de suma importancia destacar del ejemplo que, antes que en dicha raíz, la congestión aparece "localmente" en el resto de conmutadores afectados por ella. Es evidente que esta observación invalida ya claramente la idea de que el crecimiento de los árboles se produce siempre desde su raíz hacia las hojas, pero en cualquier caso, el siguiente punto se ocupa precisamente de presentar con detalle distintas formas de crecimiento de árboles de congestión, ahondando en este aspecto.

5.1.1.2. Formación de árboles de congestión

Como ya se ha mencionado varias veces en este trabajo (ver, por ejemplo, la sección 2.6.2), la congestión, tras aparecer en un determinado punto de la red, tiende a propagarse por ella, en forma de árboles de congestión cuyas ramas se extienden por varios conmutadores y enlaces.

[3]En lo sucesivo, asumiremos que todos los conmutadores mostrados en las figuras que acompañan este estudio sobre la evolución de los árboles de congestión son de tipo *CIOQ*.

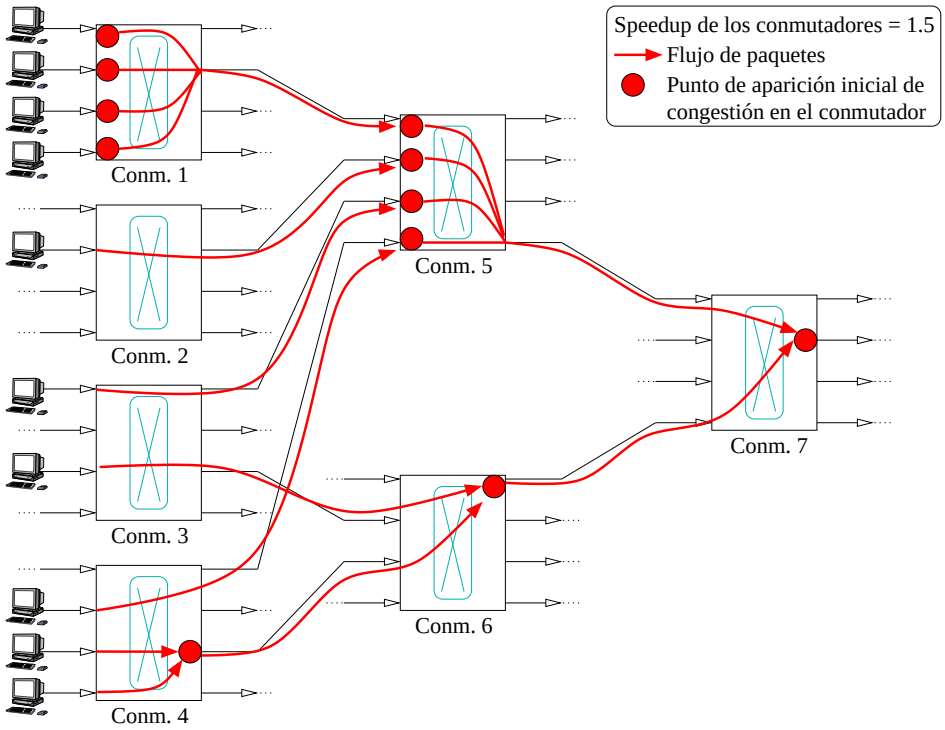


Figura 5.2: Aparición de congestión en entradas o salidas de distintos conmutadores de una red.

En la sección anterior ya se ha apuntado que la formación de árboles de congestión no se ajusta necesariamente a la concepción "clásica" del crecimiento de árboles, según la cual este crecimiento se produciría siempre desde un punto congestionado (raíz) hacia las fuentes emisoras de los flujos de paquetes responsables de dicha congestión (esto es, hacia las hojas del árbol, en sentido contracorriente o *"upstream"*). Obviamente, esto no significa que este tipo de crecimiento no sea posible. Un ejemplo de árbol que se formará de esta manera puede observarse en la figura 5.3, donde se ha asumido un valor de *speedup* de los conmutadores de 1,5, y también una inyección de las fuentes al 100 % del ancho de banda del enlace. Como puede verse, los dos flujos de paquetes confluyen en un conmutador, donde aparece un punto congestionado en una de sus salidas. En este caso el propio control de flujo propagará la congestión efectivamente en sentido "*upstream*". Es decir, las colas en las entradas y salidas por donde crucen los flujos se irán saturando sucesivamente, siempre en dicho sentido, de modo que una cola se saturará antes cuando más próxima esté a la raíz. Finalmente, si la duración de los flujos es la suficiente, la congestión alcanzará las propias fuentes.

Este tipo "clásico" de crecimiento, aún siendo posible, es ciertamente improbable en redes de cierto tamaño si los flujos implicados en la congestión son numerosos. Esto es debido a que basta con que algunos de estos flujos confluyan en un conmutador antes de alcanzar la raíz, para que la congestión aparezca antes en dicho conmutador (bien en las entradas, bien en las salidas) que en aquél donde se encuentra la raíz.

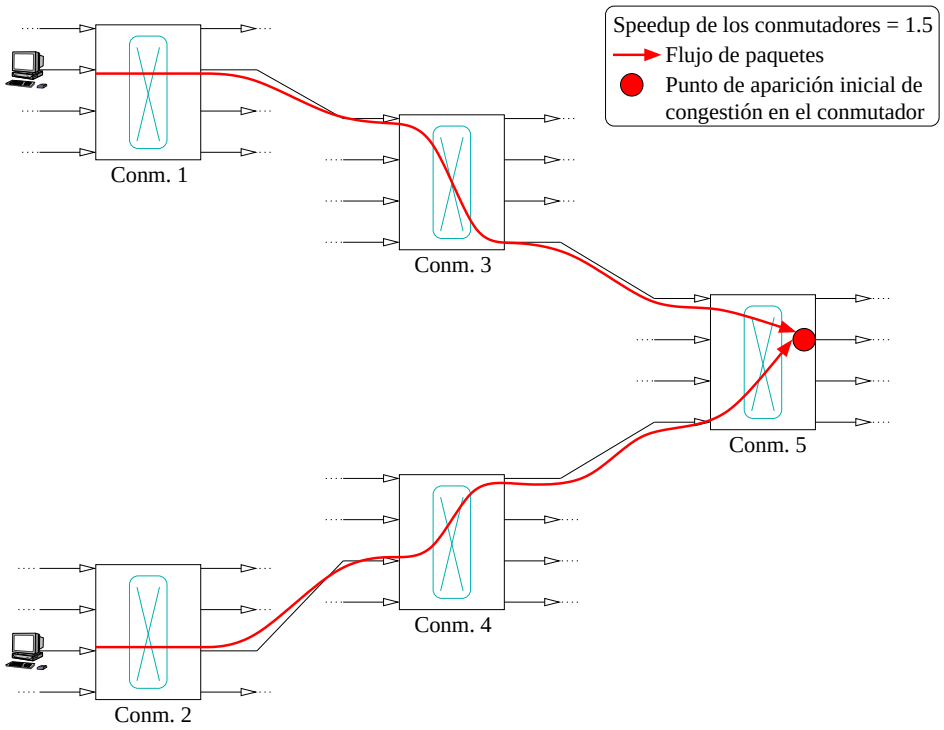


Figura 5.3: Ejemplo de árbol de congestión que crece desde la raíz hasta las hojas.

Por tanto, en estos casos, la congestión no se expandirá ordenadamente desde la raíz hacia las hojas, sino que aparecerá antes en uno o varios puntos intermedios. Este caso puede comprobarse perfectamente volviendo a revisar la figura 5.2, donde, como se ha comentado, los puntos de congestión de los conmutadores 1, 4, 5 y 6 aparecen antes que el del conmutador 7, que al final resulta ser la raíz del árbol. Dicho de otro modo, en este caso, en la red van apareciendo varios árboles "temporales" (con sus correspondientes raíces), que acaban siendo subárboles de uno mayor. En conclusión, en este caso, el árbol no se forma en sentido "*upstream*", sino al contrario.

Ahora bien, en los ejemplos anteriores se asumía que todos los flujos implicados en la congestión comenzaban a inyectarse al mismo tiempo, pero esto no tiene por qué ser así, evidentemente. Al considerar que estos flujos pueden aparecer en distintos instantes, tendremos nuevas variantes en el modo en que los árboles evolucionan.

Por ejemplo, la figura 5.4 muestra una situación donde los flujos congestionados se inyectan en diferentes instantes (en la figura, los flujos inyectados primero aparecen en línea continua, mientras que los flujos inyectados a posteriori aparecen en línea discontinua). De nuevo, se asume un valor de *speedup* de 1,5 en todos los conmutadores y una inyección desde las fuentes al 100 % del ancho de banda del enlace. Como puede verse en la figura 5.4.a, en un primer momento ($t0$), existen cuatro flujos que confluyen en el mismo conmutador (conmutador 1), formando un único flujo que continúa su camino a través de la red. En ese instante, existe un árbol de congestión cuya raíz se

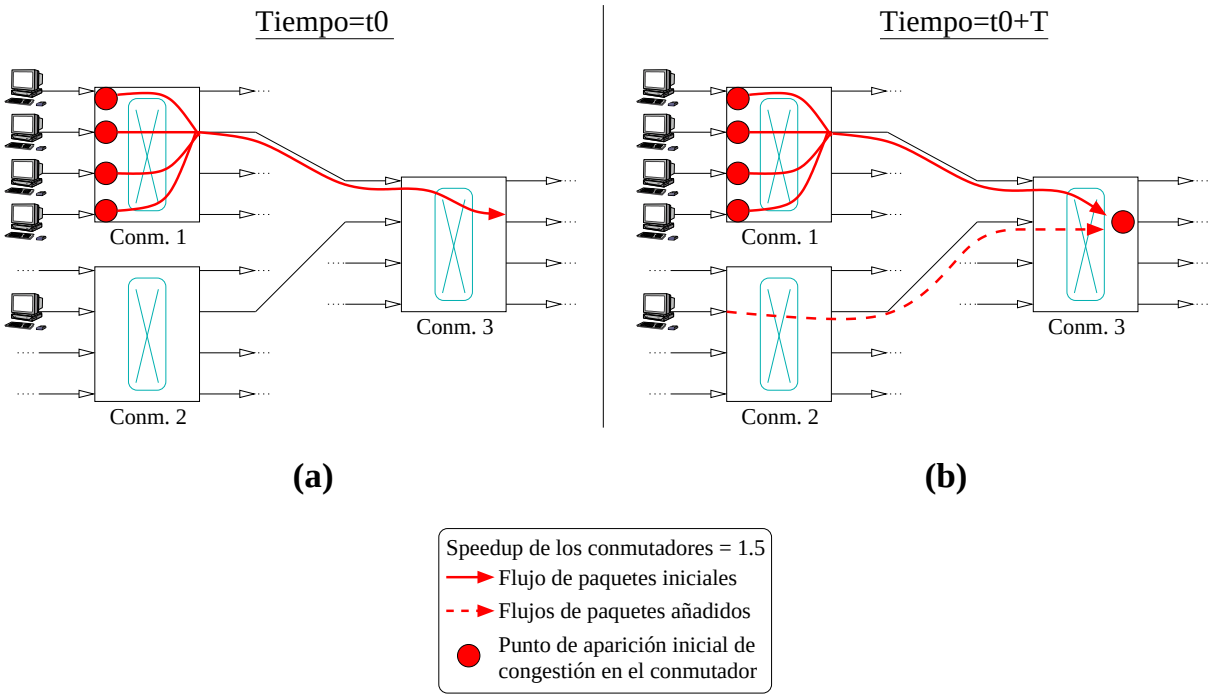


Figura 5.4: Ejemplo de árbol de congestión cuya raíz se desplaza en sentido "*downstream*".

encuentra precisamente en el conmutador 1. Al añadir un nuevo flujo en el instante $t0 + T$ (figura 5.4.b), su confluencia con el flujo resultante de los cuatro iniciales en el conmutador 3 hace que aparezca un nuevo punto de congestión, que realmente es la raíz de un nuevo árbol más extenso que el original (el cual se convierte, obviamente, en un subárbol del árbol final). Puesto que la raíz de este nuevo árbol se encuentra en sentido "*downstream*" respecto de la raíz del primer árbol, este tipo de evolución puede contemplarse como un movimiento de la raíz en sentido "*downstream*". Esto contradice, de nuevo, la idea del crecimiento "*upstream*" como única posibilidad de evolución de los árboles de congestión.

Este mismo tipo de evolución puede deberse no sólo al añadido de flujos independientes a un árbol ya existente, como en el ejemplo anterior, sino que puede deberse también a la convergencia de dos árboles de congestión para formar un único árbol. La figura 5.5 muestra un ejemplo de este caso, donde de nuevo se asume un *speedup* de 1,5 en todos los conmutadores. Además, se asume de nuevo que la inyección de los flujos implicados no es simultánea, aunque siempre se realiza al 100 % de la velocidad permitida por el enlace. En concreto, una serie de flujos se inyectan en primer lugar (los representados en línea continua), formando un árbol de congestión cuya raíz se sitúa en el conmutador 5. Más adelante, se inyecta otra serie de flujos (los representados en línea discontinua), que forman, independientemente de los flujos anteriores, un nuevo árbol cuya raíz se sitúa en el conmutador 6. Posteriormente, los flujos resultantes de ambos

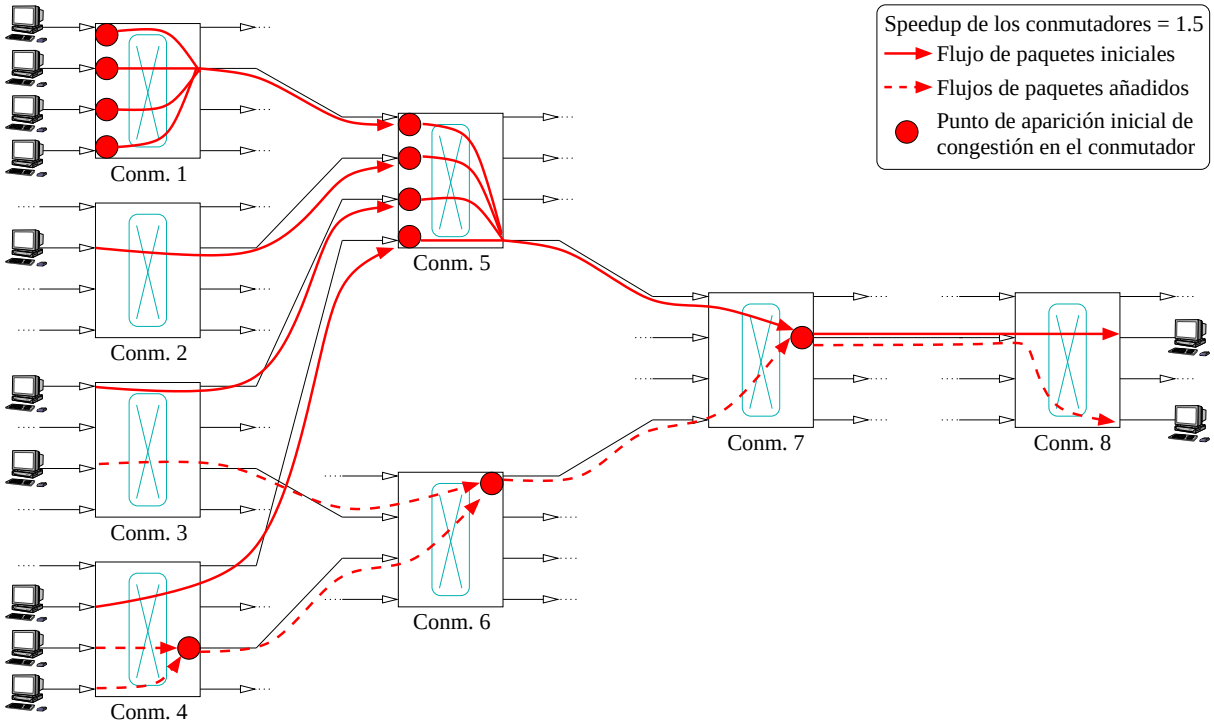


Figura 5.5: Ejemplo de árbol de congestión formado por la convergencia de dos árboles de congestión independientes.

árboles confluyen en el conmutador 7, donde se forma un nuevo punto de congestión que se convierte en la raíz de un único árbol que engloba a los dos árboles anteriores. Nótese que, efectivamente, esta nueva raíz aparece con posterioridad a las raíces de los dos árboles originales: en el caso de la raíz del primer árbol, esto es obvio, pues dicha raíz se forma antes de inyectarse la segunda tanda de flujos, y en el caso de la raíz del segundo árbol, los flujos que la forman confluyen antes en el conmutador 6 que los flujos que confluyen en el conmutador 7 para formar la raíz definitiva. Por tanto, puede decirse que, para los dos árboles originales, se produce un movimiento de la raíz en sentido "*downstream*".

Por otra parte, es interesante destacar que para que se produzca esta convergencia de dos árboles en uno, no es imprescindible que los paquetes de los flujos que forman dichos árboles tengan un mismo destino final. Precisamente en el ejemplo de la figura 5.5, puede verse que la nueva raíz del árbol se encuentra en un puerto interno de la red, donde todos los flujos implicados coinciden, siendo irrelevante, de cara a la aparición de este punto de congestión, si posteriormente los flujos se separan o no hacia destinos finales diferentes.

Sin embargo, la coincidencia e interacción de dos árboles de congestión en algún punto de la red no garantiza que dichos árboles converjan en uno, sino que es posible que dos árboles de congestión se solapen parcialmente en la red sin mezclarse. La figura 5.6

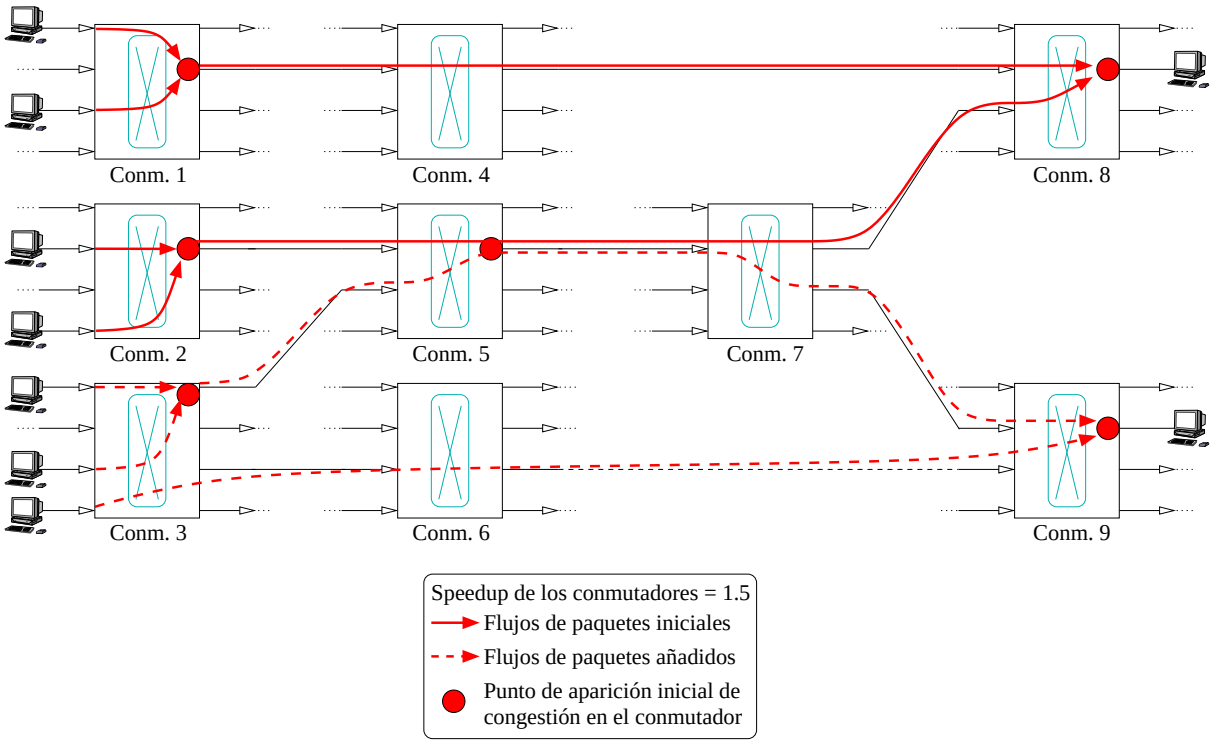


Figura 5.6: Ejemplo de árboles de congestión independientes en una misma red que se solapan sin mezclarse.

(donde se asumen las mismas condiciones respecto a *speedup* y tasa de inyección que en los ejemplos anteriores) muestra una de estas situaciones. Puede observarse que existen dos árboles de congestión independientes (uno inyectado primero, y el otro a posteriori), y que ramas de ambos árboles coinciden en cierto tramo de la red. Concretamente, la parte común es el enlace entre los conmutadores 5 y 7, con los correspondientes *buffers* a ambos extremos del enlace. A pesar de que, efectivamente, aparece un nuevo punto de congestión en la red en el conmutador 5, no se produce una fusión real de ambos árboles, ya que cada uno de ellos tiene su propia raíz (en los conmutadores 8 y 9, respectivamente) y ninguna de ellas es el punto de congestión común. Es decir, en este caso, y a pesar de la interacción, ambos árboles se forman independientemente uno de otro. Es importante resaltar que el punto de congestión común puede considerarse como perteneciente a ambos árboles, lo cual, como se verá, supone un problema a resolver si se pretende distinguir con exactitud los paquetes pertenecientes a cada árbol (como *RECN* pretende, por otra parte).

En definitiva, todos estos ejemplos nos muestran que los árboles de congestión pueden formarse en sentido "*downstream*" o en sentido "*upstream*", por diversas causas, y además, es posible que un árbol se forme a partir de la unión de varios árboles, aunque ésto no se producirá siempre que dos árboles coincidan en un punto. Esta variedad en la formación de árboles puede afectar a la efectividad de los mecanismos de control de

congestión si no es tenida en cuenta en su diseño, como comprobaremos para el caso de *RECN*. Por otra parte, puede intuirse que, análogamente, la desaparición de árboles de congestión podrá suceder de diversas formas, lo cual se detalla en el punto siguiente.

5.1.1.3. Desaparición de árboles de congestión

Obviamente, para que los árboles de congestión desaparezcan es necesario que las fuentes emisoras de los flujos que los forman dejen de inyectar paquetes, o al menos reduzcan su tasa de envío lo suficiente. De igual modo que en la formación de árboles, distintas combinaciones de arquitectura del conmutador y patrones de tráfico hacen que la desaparición de árboles pueda seguir diversos esquemas.

La idea "clásica" sobre la desaparición de los árboles se asemeja a la correspondiente sobre su formación, pero evolucionando en sentido contrario. Así, según esta idea, los árboles desaparecerían desde las hojas del árbol hasta la raíz del mismo. Es sencillo imaginar un caso de este tipo de desaparición si suponemos, por ejemplo, que en la situación mostrada en la figura 5.3, una vez formado el árbol, todas las fuentes dejan de inyectar paquetes simultáneamente. De ser así, las colas de los puertos implicados más próximos a las fuentes se vaciarán las primeras (incluso independientemente del valor de *speedup* de los conmutadores), al dejar de recibir paquetes que puedan mantener la congestión. El resto de colas de la red continuarán vaciándose sólo tras vaciarse las colas situadas en sentido "*upstream*" de las que reciben paquetes, por lo que el árbol desaparecería, efectivamente, desde la raíz hasta las hojas.

Ahora bien, este caso en el que todas las fuentes dejan de inyectar paquetes simultáneamente, aunque posible, resulta poco probable, o, al menos, demasiado singular. Una situación más normal será aquélla en la que los flujos que forman un árbol desaparezcan en instantes distintos, y esto precisamente puede dar lugar a variadas evoluciones de la congestión antes de su desaparición total.

Por ejemplo, la figura 5.7 muestra una situación en la que la desaparición del árbol no se produce desde las hojas hacia la raíz. En la figura 5.7.a se representa la situación hasta el instante $t0$: varios flujos contribuyen a mantener un árbol de congestión cuya raíz se sitúa en el conmutador 3. Se ha asumido que hasta ese instante todos los flujos se inyectan a la máxima velocidad permitida por el enlace. Ahora bien, en un instante inmediatamente posterior, uno de los flujos (el representado en azul) desaparece al dejar de inyectar paquetes su fuente correspondiente (figura 5.7.b). Esto significa que en el conmutador 3 ya no existirá ningún punto de congestión, y por tanto, la raíz del árbol original ya no estará en dicho conmutador, sino que se "desplazará" en sentido "*upstream*" (hasta el conmutador 1, de hecho). Por consiguiente, el árbol original se colapsa, en ese tramo, desde la raíz hacia las hojas. Es evidente que esto contradice la idea de una única dinámica (colapso desde las hojas hacia la raíz) en la desaparición de árboles de congestión.

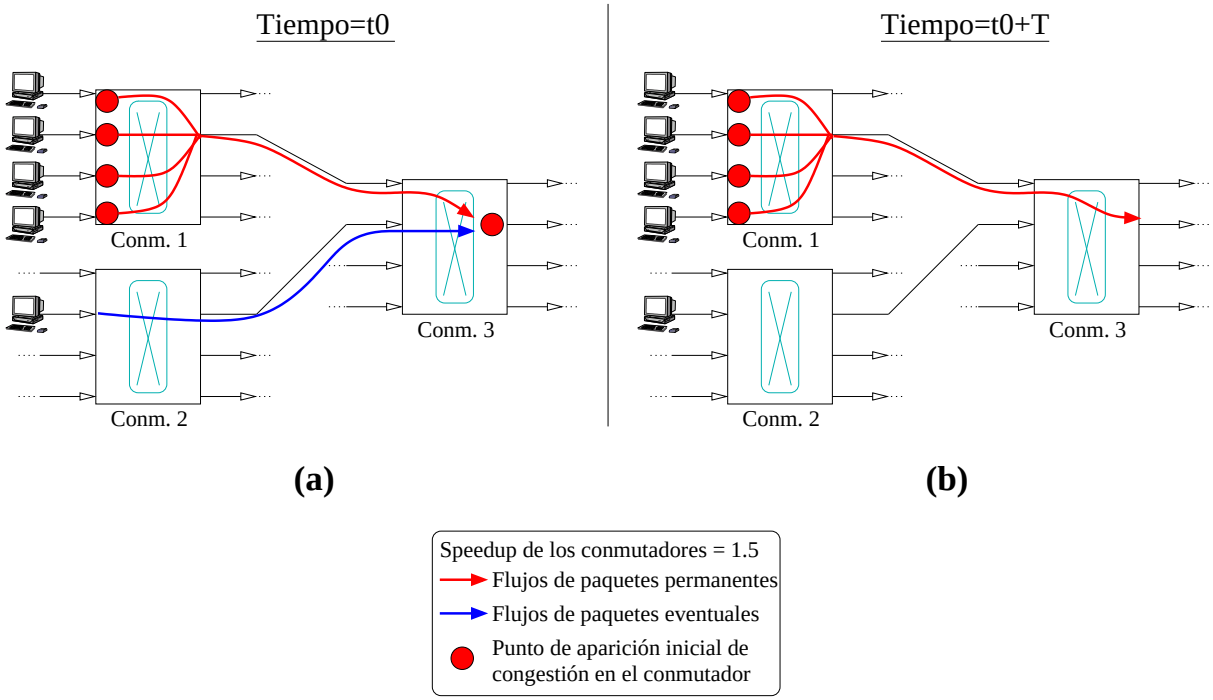


Figura 5.7: Ejemplo de desaparición parcial de árbol de congestión con desplazamiento "*upstream*" de la raíz.

Al igual que las distintas formas de crecimiento, esta variedad en la forma de desaparición de los árboles de congestión también debería ser tenida en cuenta por cualquier técnica de control de congestión que pretenda ser eficiente, especialmente si dicha técnica se basa en una asignación y liberación dinámica de los recursos destinados a combatir la congestión, como es el caso de *RECN*. De hecho, el no considerar las distintas formas de desaparición de los árboles es una de las causas de ineficiencia de la propuesta original de *RECN*, tal y como se explica en el siguiente punto.

5.1.2. Impacto en el mecanismo

El análisis precedente sobre la evolución de los árboles nos permite explicar en este punto, con todo fundamento, los problemas que algunos tipos de evolución plantean a la propuesta original de *RECN*. De cara a centrarnos rápidamente en dichos problemas, conviene presentar ya los defectos que se detectaron en el mecanismo original tras nuestro estudio sobre los árboles. Dichos defectos son los siguientes:

- **Posibles detecciones incorrectas de la raíz del árbol**.
- **Imposibilidad de adaptación al desplazamiento "*downstream*" de la raíz de un árbol**.

- **Posible consumo innecesario de *SAQs* durante la desaparición de un árbol**.

Las razones exactas de estos defectos, así como sus consecuencias (graves, en algunos casos) en la eficacia del mecanismo, se detallan en los siguientes puntos.

5.1.2.1. Detecciones incorrectas de la raíz del árbol

Este defecto es debido concretamente a que la propuesta original de *RECN* sólo detecta congestión en las salidas de los conmutadores (ver sección 4.3.1). Teniendo esto en cuenta, si la congestión aparece primero en las entradas de un conmutador (como es perfectamente posible, según se ha expuesto anteriormente), la congestión no se detectará hasta que el control de flujo propague dicha congestión a las colas situadas en las salidas de los conmutadores conectados a las entradas congestionadas. Esto implica no sólo que la congestión se detectará con retraso, sino también que el punto exacto de aparición de congestión no se detectará correctamente.

La importancia de este defecto es evidente si consideramos que, al no detectar exactamente la posición de la raíz del árbol, no es posible separar de forma totalmente correcta los paquetes de flujos congestionados (aquéllos que pasarán realmente por la auténtica raíz del árbol) de los paquetes de flujos no congestionados. Dicho de otro modo, es posible que paquetes congestionados sean considerados no congestionados, y viceversa, con lo cual puede suceder que paquetes de uno y otro tipo se almacenen en la misma cola. Naturalmente, en estos casos, los paquetes congestionados producirían *HOL blocking* sobre los no congestionados.

La figura 5.8 muestra un ejemplo donde este problema se pone de manifiesto. En la situación mostrada en la figura 5.8.a, varios flujos confluyen en el conmutador 2 solicitando la misma salida. Al tener el conmutador un *speedup* de valor 1, aparece congestión (instante $t0$) en las entradas por donde los flujos llegan al conmutador. Dicha congestión, según la propuesta original de *RECN*, no se detecta al no producirse en una salida del conmutador 2. Posteriormente, y por efecto del control de flujo, la congestión se propaga a las colas de las salidas de los conmutadores situados en sentido "*upstream*", como es el caso de la cola en la salida $P5$ del conmutador 1. Cuando el nivel de ocupación de esta cola alcance el umbral de detección (instante $t0 + T$, en la figura 5.8.a), se notificará congestión a las entradas que envían paquetes hacia dicha cola (en el ejemplo, sólo se notifica a la entrada $P1$ en el conmutador 1).

Al recibirse dicha notificación en la entrada, se asignará una *SAQ* para almacenar paquetes dirigidos al punto de congestión detectado y notificado, como puede comprobarse a partir del contenido del registro *CAM* correspondiente que aparece en la figura 5.8.b. Ahora bien, como se ha mencionado, dicho punto no es la auténtica raíz del árbol: la *SAQ* se asocia a la salida $P5$ del conmutador 1, mientras que realmente

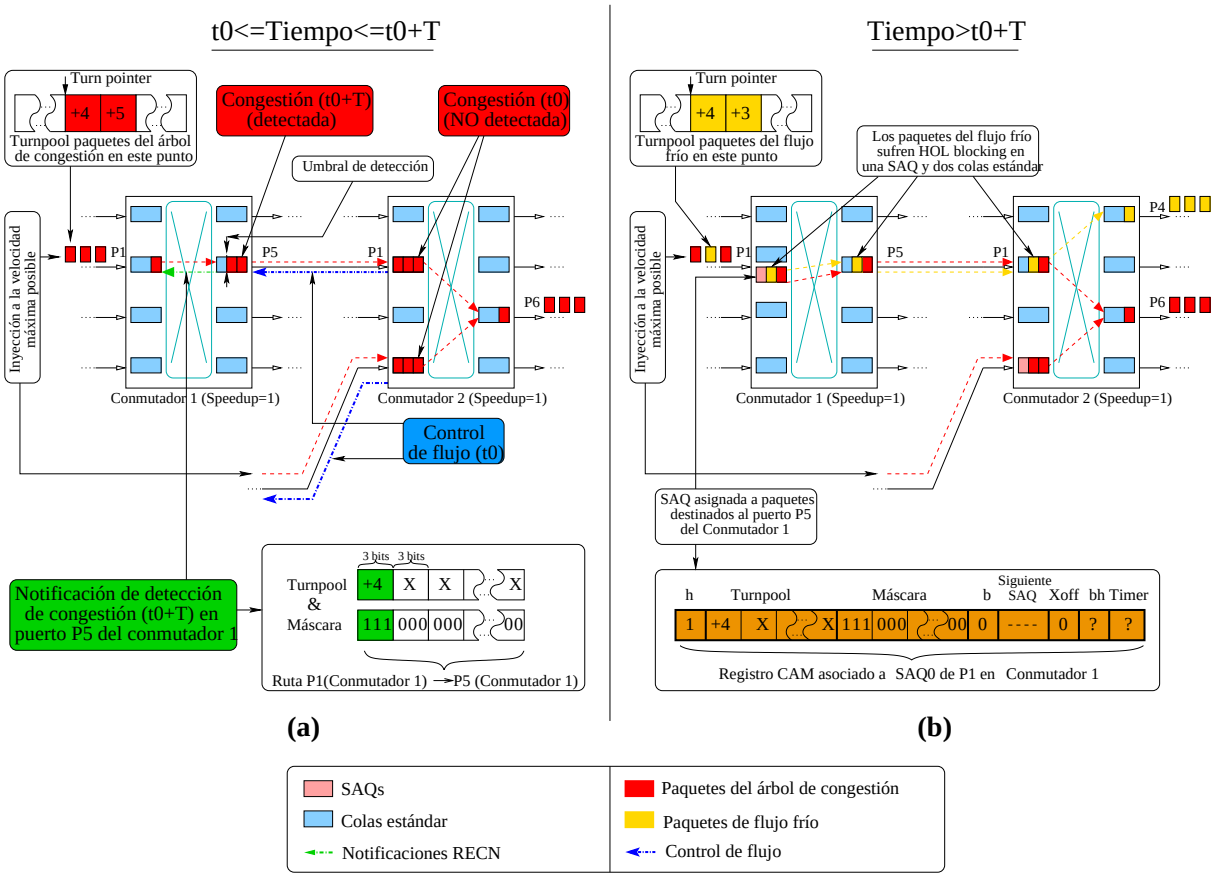


Figura 5.8: Detección incorrecta de la raíz de un árbol y consecuencias en la separación de flujos fríos y calientes.

la raíz del árbol se encuentra en la salida $P6$ del conmutador 2. En este caso, tras la asignación de la *SAQ*, un flujo de paquetes no congestionado puede confundirse con un flujo congestionado al llegar al punto donde se ha asignado la *SAQ*, siempre que dicho flujo solicite la salida erróneamente detectada como raíz del árbol. Este es el caso del flujo representado en amarillo en la figura 5.8.b, que no pasa por la raíz real del árbol, pero sí por la raíz "falsa" del mismo. Lógicamente, los paquetes del flujo "frío" se almacenarán en la *SAQ* de la entrada $P1$ del conmutador 1 junto con los paquetes congestionados, produciéndose *HOL blocking*. De forma similar, paquetes congestionados y no congestionados compartirán cola (estándar) tanto en la salida $P5$ del conmutador 1, como en la entrada $P1$ del conmutador 2, ya que en estos puntos no se han asignado *SAQs* por haberse detectado incorrectamente la raíz del árbol. En ambas colas estándar, los paquetes de flujos no congestionados sufrirán de nuevo *HOL blocking*.

En definitiva, es posible que la propuesta original de *RECN* detecte incorrectamente el punto donde aparece la congestión, lo que a su vez puede llevar, como hemos visto, a la introducción masiva de *HOL blocking*, no sólo por no asignar *SAQs* en puntos donde esto es necesario para evitar el *HOL blocking*, sino también por asignar *SAQs*

para contener paquetes dirigidos a puntos donde no se encuentra la raíz real del árbol. Evidentemente, el mecanismo debe ser corregido para que se detecten correctamente todos los puntos de aparición de congestión. En un próximo punto se describirá la solución adoptada para conseguirlo.

5.1.2.2. Falta de adaptación al desplazamiento "*downstream*" de la raíz de un árbol

Incluso aunque se detecten correctamente todos los puntos de aparición de congestión, es posible que la propuesta original de *RECN* no identifique en algunos momentos la posición auténtica de la raíz de un árbol. Esto es debido al carácter dinámico de la evolución de los árboles de congestión y a las limitaciones de la propuesta original en cuanto al tipo de notificaciones de congestión que pueden aceptarse.

Más concretamente, en el análisis sobre evolución de los árboles hemos visto que, debido a las diferentes dinámicas, la posición de raíz de un árbol puede cambiar con el tiempo. Pues bien, en el caso en que la posición de la raíz de un árbol se desplace en sentido "*downstream*", la propuesta original de *RECN* no será capaz de asimilar dicho desplazamiento (aunque la nueva posición se detecte) debido a que no se aceptan notificaciones referidas a puntos de congestión más específicos que los asociados a *SAQs* ya asignadas (ver sección 4.3.5). Nótese que al desplazarse en sentido "*downstream*" una raíz ocupará una posición más alejada respecto a todas las *SAQs* asignadas hasta entonces, y al mismo tiempo su posición será una ampliación de las rutas asociadas a dichas *SAQs* (el árbol se amplía "hacia abajo", pero no cambian sus ramas). Por tanto, las notificaciones sobre la nueva posición que lleguen a los puntos donde se encuentran dichas *SAQs* serán más específicas y, en consecuencia, se descartarán.

En estos casos, las *SAQs* asignadas originalmente para contener paquetes del árbol permanecerán asociadas a puntos donde ya no se encuentra realmente la raíz y, lo que es peor, no se asignarán en los puertos notificados nuevas *SAQs* que se asocien a la nueva posición de la raíz. En estas situaciones tendremos el mismo riesgo que en el caso de detección incorrecta de la raíz: al estar equivocado el mecanismo sobre la auténtica posición de ésta, los paquetes congestionados pueden confundirse con los no congestionados, y podrían almacenarse paquetes de ambos tipos en la misma cola con la consiguiente aparición del *HOL blocking*.

Es muy importante destacar que la nueva raíz se detectará, en cualquier caso, como nuevo punto de congestión, y que en algunos puntos del árbol las notificaciones sobre la nueva raíz pueden aceptarse. Concretamente, esto será así en aquellos puntos en los que anteriormente, por alguna circunstancia, no se hubieran asignado *SAQs* para contener paquetes del árbol. En estos casos, puede decirse que la propuesta original de *RECN* detectaría como varios árboles con distintas raíces aquello que en realidad es

un único árbol de congestión. Obviamente, esto no resta peligro a este problema, pues seguirán existiendo *SAQs* asociadas a posiciones "falsas" de la raíz.

La figura 5.9 muestra un ejemplo de la falta de adaptación de la propuesta original de *RECN* al movimiento "*downstream*" de la raíz del árbol y de las consecuencias negativas que esto acarrea. En la figura, se asume que los conmutadores tienen un valor de *speedup* de 2. Además, se asume que los flujos representados con línea discontinua se inyectan desde un instante anterior al instante $t0$, y que los flujos representados con línea continua aparecen inmediatamente después del instante $t0$.

Teniendo esto en cuenta, la figura 5.9.a refleja la detección de dos puntos diferentes de congestión en la red representada, en instantes distintos. El primero de estos puntos de congestión aparece en la salida $P5$ del conmutador 1, debido a la confluencia de los dos flujos inyectados previamente. Como indica la figura, dicho punto de congestión se detecta en el instante $t0$, y esto conlleva la notificación de esta detección a las entradas que envían paquetes hacia dicha salida. En ese momento, dicho punto es la raíz del único árbol de congestión presente en la red representada, pues hasta entonces al conmutador 2 sólo llega un flujo de paquetes, por lo que en dicho conmutador no existe congestión. Ahora bien, dos nuevos flujos llegan inmediatamente después al conmutador 2, contendiendo con el ya existente por la misma salida ($P6$), que se convierte de este modo en un punto de congestión. Esta congestión se detecta en el instante $t0 + T$, y como tal es comunicada a las entradas del mismo conmutador que contribuyen a la congestión. Nótese que, tras $t0 + T$, en la red existe realmente un único árbol de congestión, cuya raíz es la salida $P6$ del conmutador 2, lo que puede contemplarse como un desplazamiento "*downstream*" de la raíz del árbol anterior.

Sin embargo, la nueva posición de la raíz no se refleja en todas las *SAQs* que se asignan para contener paquetes del árbol. Al haberse detectado antes la congestión en la salida $P5$ del conmutador 1, las *SAQs* asignadas en las entradas de dicho conmutador tras las correspondientes notificaciones están asociadas, lógicamente, a dicho punto, y no a la nueva raíz del árbol. Esto puede comprobarse en la figura 5.9.b, donde aparece el registro *CAM* correspondiente a la *SAQ* asignada en la entrada $P1$ del conmutador 1. Por contra, las *SAQs* asignadas en las entradas del conmutador 2 tras la detección del segundo punto de congestión sí están asociadas a la raíz auténtica del árbol. Esto también puede apreciarse en la figura 5.9.b, comprobando el registro *CAM* correspondiente a la *SAQ* asignada en la entrada $P1$ del conmutador 2.

También puede comprobarse en la figura 5.9.b que, tal y como está establecido, no se asignan *SAQs* en las salidas donde se detecta congestión ($P5$ del conmutador 1 y $P6$ del conmutador 2). Por otra parte, si la ocupación de las *SAQs* asignadas alcanza el umbral de propagación, se notificará congestión en sentido "*upstream*". Por ejemplo, en la misma figura puede apreciarse que esto sucede con la *SAQ* asignada en la entrada $P1$ del conmutador 2, que notifica congestión precisamente a la salida $P5$ del conmutador

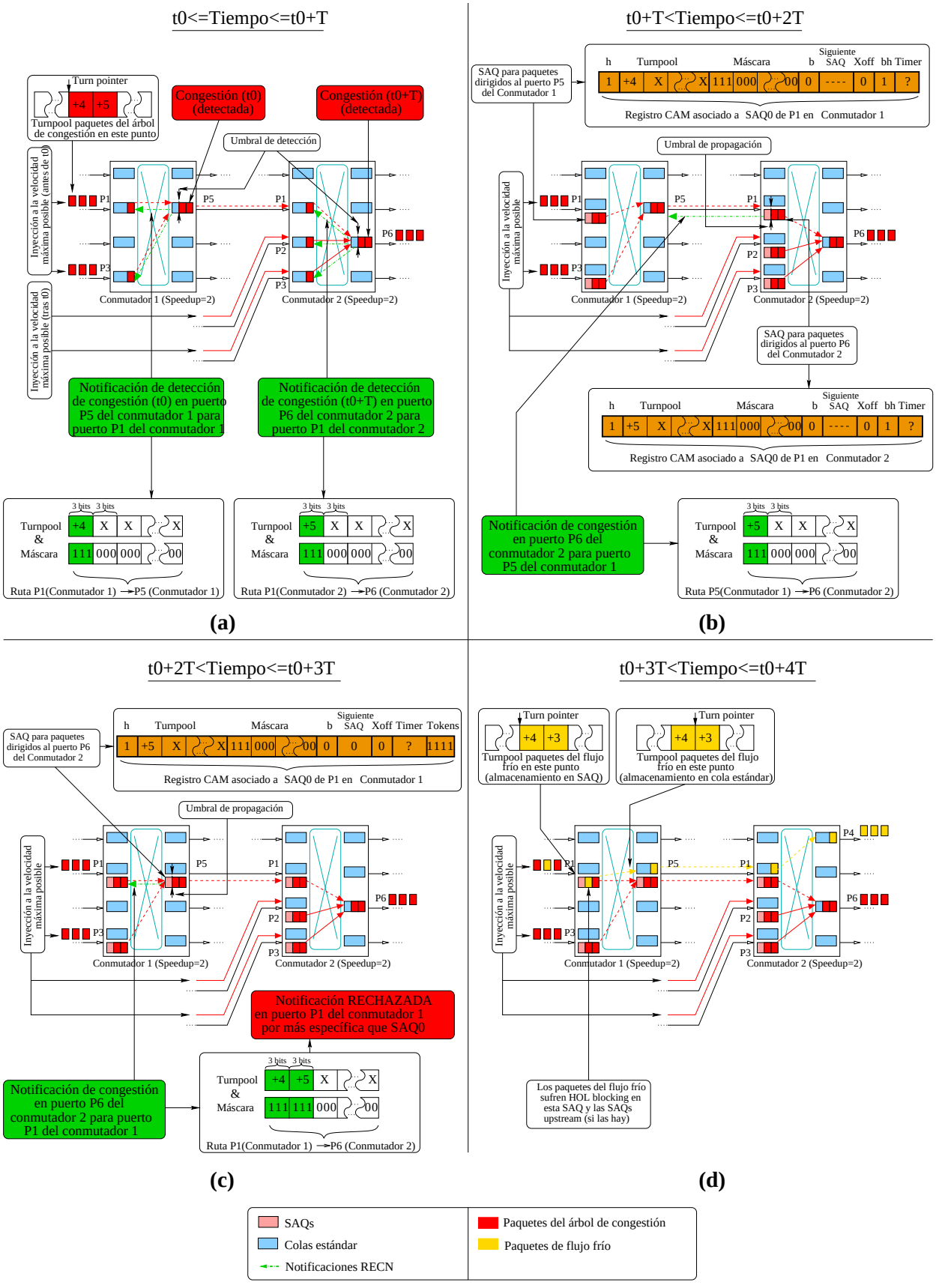


Figura 5.9: Detección de un único árbol de congestión como dos árboles distintos y consecuencias en la separación de flujos fríos y calientes.

1. Al no haber ninguna *SAQ* asignada en este punto, dicha notificación será aceptada sin problemas, asignándose consecuentemente una *SAQ*.

El registro *CAM* correspondiente a dicha *SAQ* puede apreciarse en la figura 5.9.c. Como puede comprobarse, el punto asociado a esta *SAQ* es, lógicamente, la salida $P6$ del conmutador 2. Si el nivel de ocupación de esta *SAQ* llega al umbral de propagación, se notificará congestión a las entradas que envíen paquetes del árbol. Pero en esta ocasión, si se notifica a las entradas $P1$ o $P3$, el punto de congestión notificado (salida $P6$ del conmutador 2) resulta ser más específico que el asociado a las *SAQs* ya asignadas en dichas entradas (salida $P5$ del conmutador 1). En consecuencia, estas notificaciones se rechazarán, lo cual es grave si consideramos que el punto notificado es la auténtica raíz del árbol.

La gravedad de este rechazo se pone de manifiesto si consideramos la situación mostrada en la figura 5.9.d, que representa un instante posterior en el que un nuevo flujo de paquetes (representado en amarillo) aparece en escena. Dicho flujo pasa por la salida $P5$ del conmutador 1, pero no por la raíz real del árbol. Puede comprobarse que en la entrada $P1$ del conmutador 1, donde existe una *SAQ* asociada al punto $P5$ del conmutador 1, los paquetes de este flujo "frío" se almacenarán en dicha *SAQ*, compartiendo esta cola con los paquetes congestionados que sí pasarán por la raíz real del árbol de congestión. Esto producirá *HOL blocking* en este punto, evidentemente. Puede apreciarse que tanto en la salida $P5$ del conmutador 1 como en la entrada $P1$ del conmutador 2, los paquetes del flujo frío son correctamente separados de los paquetes congestionados (los paquetes fríos se almacenan en la cola estándar), ya que en dichos puntos las *SAQs* sí están asociadas a la raíz real del árbol, y por tanto en estos puntos no se introduce *HOL blocking*.

Si se consideran los puntos a los que están asignadas las *SAQs* que aparecen en la situación final mostrada en la figura 5.9.d, se aprecia claramente que, aunque en la red existe un único árbol de congestión, el mecanismo establecido por la propuesta original de *RECN* ha acabado detectándolo como dos árboles distintos y contiguos. Nótese que realmente, uno de ellos (aquél cuya raíz es "falsa") no es sino una rama del árbol real. Además, dados los rechazos de notificaciones más específicas, la percepción del mecanismo respecto a este árbol no variará a menos que se produzca algún cambio en el tráfico (desaparición o añadido de flujos). Evidentemente, este fenómeno puede darse en mayor grado en redes mayores, donde un único árbol de congestión real podría detectarse como multitud de subárboles.

La figura 5.10 muestra precisamente un ejemplo de cómo detectaría la propuesta original de *RECN* un único árbol de congestión real cuyas ramas discurren por cinco etapas de una red. En la figura se asume que todas las fuentes comienzan la inyección simultáneamente, y que el *speedup* de los conmutadores es 1,5. En la figura se distingue cada árbol detectado mediante un límite representado en línea discontinua, dentro del cual se encuentran los puntos (entradas o salidas) por donde se considera que discurre

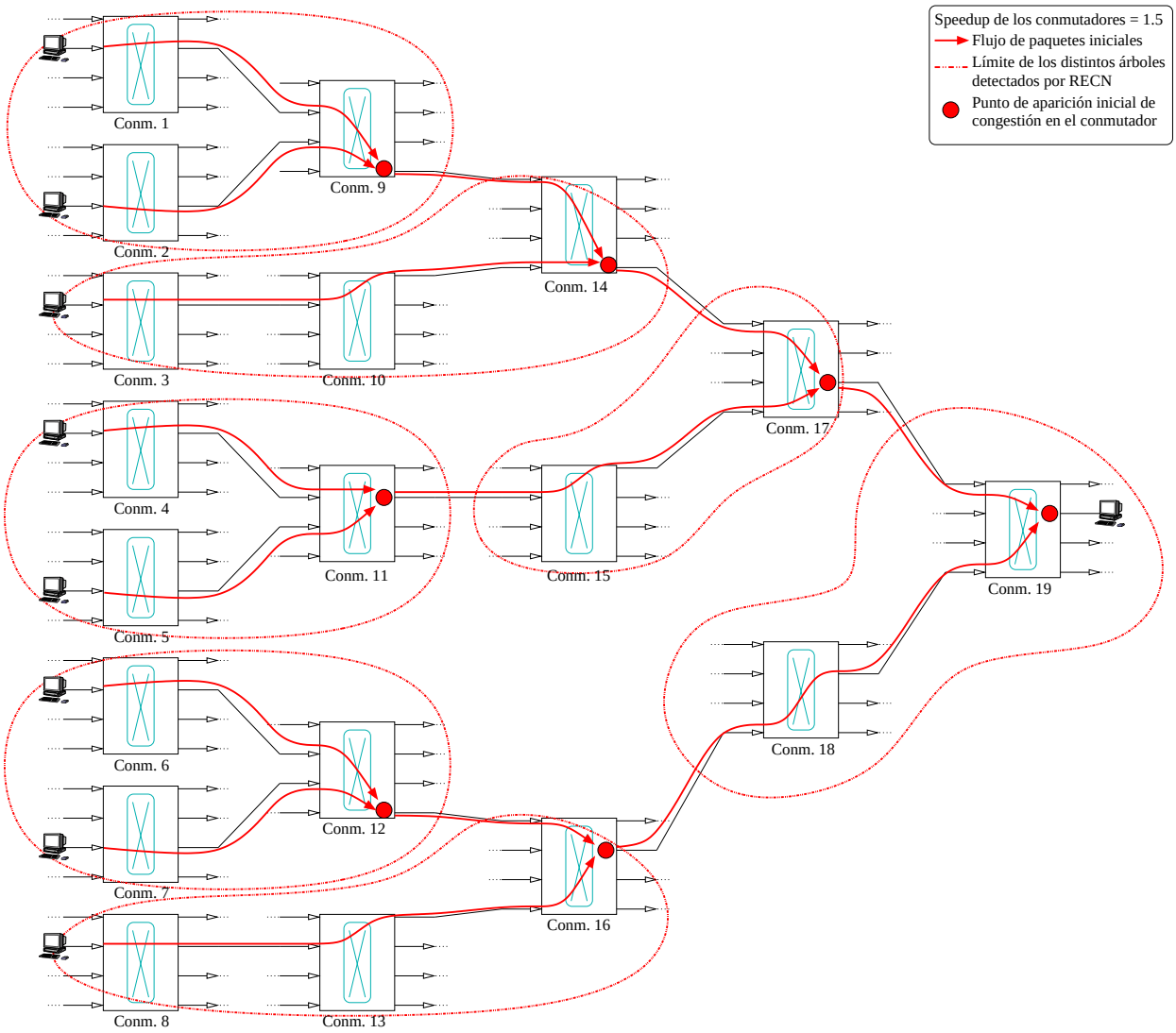


Figura 5.10: Detección de un único árbol de congestión como varios árboles distintos.

el árbol de congestión en cuestión. Puede observarse que un único árbol ha acabado siendo detectado como siete árboles distintos. Cabe recordar (ver sección 5.1.1.2) que no es necesario que se añadan flujos para que la congestión aparezca antes en puntos situados en las primeras etapas, sino que basta con que los flujos confluyen antes en éstas, tal y como sucede en este caso. Es sencillo imaginar, extrapolando la situación mostrada en la figura 5.9.d a este nuevo ejemplo, que la cantidad de puntos donde puede producirse *HOL blocking* es mucho mayor, por lo que el problema será mucho más grave.

En conclusión, también el desplazamiento "*downstream*" de la raíz de un árbol de congestión puede llevar a que la propuesta original de *RECN* no sea capaz de eliminar el *HOL blocking* de forma adecuada, al no adaptarse convenientemente a la nueva situación por no aceptar notificaciones de puntos de congestión más específicos. Dicho

de otro modo, la propuesta original de *RECN* no es eficaz si el árbol de congestión no crece desde su raíz hacia las hojas. Obviamente, esto contradice el objetivo, en cuanto a eliminación eficiente del *HOL blocking* del mecanismo, que debe modificarse para ser capaz de adaptarse dinámicamente a cambios de posición de la raíz. Las modificaciones pertinentes en este sentido se detallarán posteriormente.

5.1.2.3. Consumo innecesario de *SAQs* durante la desaparición de un árbol

Al igual que, como se ha explicado en el punto anterior, ciertos tipos de crecimiento de los árboles de congestión suponen una pérdida de eficacia de la propuesta original de *RECN*, también ciertos tipos de desaparición de árboles plantearán problemas. Si en el caso de la formación de árboles se asumía (erróneamente) que éstos siempre crecían desde la raíz hacia las hojas, en el caso de la desaparición de árboles se asume que ésta siempre se produce desde las hojas hacia la raíz, lo cual, como se ha explicado anteriormente, no es correcto.

Esta última asunción se refleja, concretamente, en el modo en que se lleva a cabo la liberación de *SAQs* una vez que la congestión se desvanece (ver sección 4.3.6). El sistema de liberación establecido por la propuesta original se basa en notificaciones de liberación sucesivas, enviadas siempre desde las hojas del árbol en la dirección de la raíz, de forma que las *SAQs* se liberan siempre ordenadamente en ese sentido. Dicho de otro modo, ninguna *SAQ* se libera si existen *SAQs* asignadas al mismo árbol y situadas en puntos "*upstream*".

Ahora bien, como hemos visto, un árbol de congestión puede desaparecer desde la raíz hacia las hojas. En esos casos, la congestión desaparecerá antes de los puntos más cercanos a la raíz original. Esto quiere decir que la ocupación de las *SAQs* situadas en estos puntos será mínima cuando no nula, y de hecho su activación no tiene sentido al no existir congestión en dichos puntos. Ahora bien, estas *SAQs* "infrautilizadas" no podrán liberarse dado que la congestión aún puede estar presente en algún punto "*upstream*" con *SAQs* asignadas para el mismo árbol. Esto es, pueden existir *SAQs* que deberían liberarse pero que no se liberan por no considerarse que en su posición se encuentre una hoja del árbol.

A priori, podría parecer que esto no supone un problema tan grave como los anteriores, en los que se introducía *HOL blocking*. Pero cabe recordar que *RECN* no pretende ser sólo eficaz, sino también eficiente y escalable, es decir, pretende ser eficaz consumiendo un número reducido y constante de recursos. Evidentemente, el mantener durante cierto tiempo *SAQs* activas cuando esto no es realmente necesario para eliminar el *HOL blocking* contradice este objetivo principal del mecanismo. Más aún, dado que el número máximo de *SAQs* activas por grupo está limitado, podría darse el caso de que en algún momento y en algún punto de la red no hubiera suficientes *SAQs* para todos los árboles de congestión realmente presentes en la red, mientras que

al mismo tiempo podrían existir *SAQs* mantenidas innecesariamente activas asignadas a un punto de congestión obsoleto. En estos casos, la estricta liberación ordenada de *SAQs* repercutiría negativamente en las prestaciones del mecanismo al impedir un uso eficiente de los recursos.

La figura 5.11 muestra un ejemplo de este uso ineficiente de *SAQs* durante la desaparición de un árbol. La situación inicial es la mostrada en la figura 5.11.a, donde varios flujos forman un árbol de congestión cuya raíz se sitúa en el puerto $P6$ del conmutador 3. Se asume que el proceso de formación del árbol sucedió en algún instante bastante anterior, así como los procesos de detección de congestión y propagación de la misma. Igualmente se asume que estos últimos procesos, de un modo u otro, se realizaron correctamente, de modo que las *SAQs* representadas en la figura se encuentran correctamente asignadas para contener paquetes dirigidos a la raíz del árbol.

En esta situación inicial, el uso de los enlaces por donde circulan los paquetes de los flujos congestionados viene determinado por el uso del enlace de salida donde se sitúa la raíz del árbol. Asumiendo que este uso sea del 100 %, y un reparto equitativo del uso del *crossbar*, el uso del resto de enlaces implicados, en ese instante, es el mostrado en la figura: cada porcentaje resulta de dividir el uso de un canal de salida por el número de flujos que solicitan dicha salida.

Por otra parte, tal y como establece la propuesta original (y como puede comprobarse en la figura 5.11.a para la *SAQ* en la entrada $P1$ del conmutador 1) las *SAQs* situadas en las hojas del árbol tienen activo el bit de hoja (o todos los bits de su lista de *tokens* si las hojas estuvieran en una salida) en su registro *CAM* correspondiente. Esto es una condición necesaria para su liberación, que podrá realizarse si se cumplen el resto de condiciones (la *SAQ* se vacía y ha expirado el *timer* desde su asignación). No sucede lo mismo para *SAQs* situadas en puntos "*dowstream*" del árbol, en cuyo caso el no tener el bit de hoja activo (entradas) o todos los bits de su lista de *tokens* activos (salidas, como es el caso, por ejemplo, de la *SAQ* en la salida $P5$ del conmutador 1, cuyo registro *CAM* se muestra en la figura) bloquea totalmente su liberación, incluso cumpliendo el resto de condiciones.

La figura 5.11.b representa un instante posterior, en el que ha cesado la inyección del flujo de paquetes situado en la rama superior del árbol (se asume simplemente que la fuente de dicho flujo deja de inyectar paquetes). Poco tiempo después de cesar la inyección, la *SAQ* situada en la entrada $P1$ del conmutador 1 se vaciará, y por tanto (y asumiendo que el *timer* desde su asignación ha expirado) se liberará, al tener activo el bit de hoja. Esta liberación se comunicará a la inmediata *SAQ* "*downstream*" asignada para el mismo árbol (o sea, a la *SAQ* situada en la salida $P5$ del conmutador 1), que, como puede comprobarse en la figura, activará el bit asociado a $P1$ en la lista de *tokens* del registro *CAM* correspondiente. Al tener ahora activos todos los bits de la lista de *tokens*, la *SAQ* en la salida $P5$ del conmutador 1 se convertirá en una hoja del árbol.

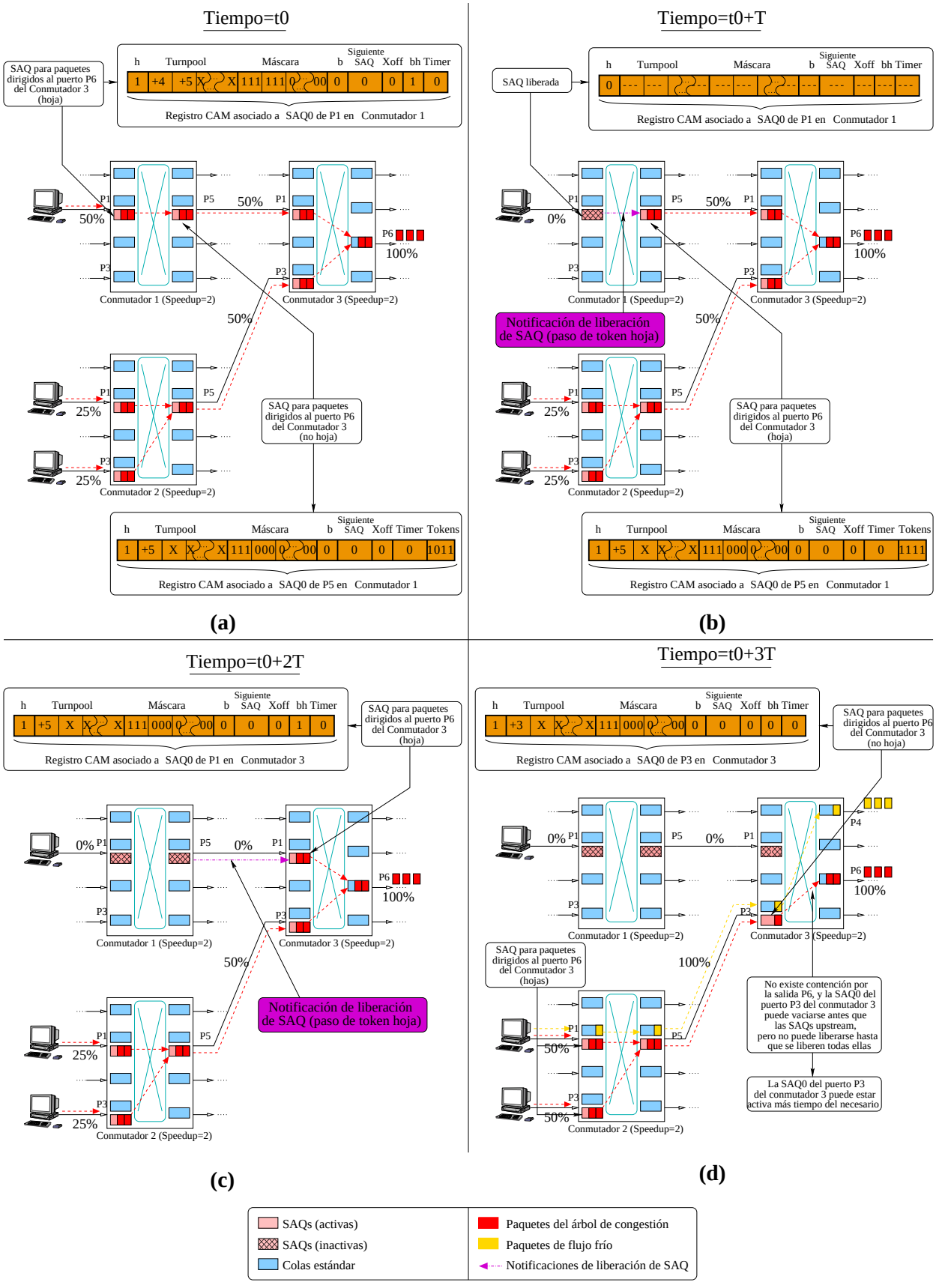


Figura 5.11: Ejemplo del uso ineficiente de SAQs tras la desaparición parcial de un árbol de congestión.

A su vez, dicha *SAQ* acabará vaciándose y liberándose en un instante posterior (figura 5.11.c), pasando a ser la *SAQ* en la entrada $P1$ del conmutador 3 la nueva hoja del árbol. Nótese que hasta ese instante, existen dos flujos solicitando la salida $P6$ en el conmutador 3, por lo que, aunque una de las fuentes de flujos congestionados haya cesado la inyección, la congestión está aún presente en dicho conmutador y en dicha salida se sitúa todavía la raíz de un árbol.

Ahora bien, cuando la *SAQ* en la entrada $P1$ del conmutador 3 se vacíe totalmente y se libere (figura 5.11.d), no habrá congestión real en el conmutador 3, pues sólo un flujo (desde la entrada $P3$) solicita la salida $P6$, que por tanto deja de ser la raíz real del árbol. En esas condiciones, el uso de los enlaces cambiará, pues la salida $P6$ del conmutador 3 ya no necesita repartir el ancho de banda de su enlace, que estará totalmente disponible para los paquetes procedentes de la entrada $P3$. En estas circunstancias, la *SAQ* en dicha entrada podría vaciarse rápidamente, o al menos más rápido que las *SAQs* en las entradas $P1$ y $P3$ del conmutador 2 (por ejemplo, si un nuevo flujo frío ocupa el enlace de entrada a $P3$ del conmutador 3, como está reflejado en la figura 5.11.d). De este modo, la *SAQ* en la entrada $P3$ del conmutador 3 puede estar vacía antes que el resto de *SAQs* del árbol, pero no podrá liberarse por no tener activo el bit de hoja. De hecho, dicha *SAQ* no se liberará hasta que no lo hagan el resto de *SAQs* del árbol y reciba una notificación de liberación desde la *SAQ* en la salida $P5$ del conmutador 2. Durante todo este tiempo, la *SAQ* en la entrada $P3$ del conmutador 3 estará asignada inútilmente, para un árbol de congestión que ya no existe en ese punto, lo que implica un uso poco eficiente de las *SAQs*. Si además hubiera carencia de *SAQs* para contener paquetes de otros árboles, se estarían utilizando incorrectamente los recursos destinados a eliminar el *HOL blocking*.

En resumen, la propuesta original de *RECN* también presenta problemas si la desaparición real de un árbol, o de parte de él, no se produce desde las hojas hacia la raíz, pudiendo en estos casos mantener recursos ocupados más tiempo del necesario debido al mecanismo de liberación estrictamente ordenada que dicha propuesta establece. Como en el caso de los problemas explicados en anteriores puntos, este defecto debe ser corregido para que la eficiencia del mecanismo no se resienta, y de modo que se adapte a cualquier tipo de desaparición de árboles de congestión. En este sentido, existe una posible solución cuya explicación se incluye en un punto de la siguiente sección.

5.2. Mejoras introducidas

Una vez identificadas con exactitud las razones concretas de la falta de eficiencia que presenta la propuesta original de *RECN*, es momento de presentar las modificaciones que deben realizarse en el mecanismo de cara a solventar los defectos detectados.

Teniendo en cuenta el análisis de los problemas del mecanismo expuesto en la sección anterior, es inmediato concluir que las distintas mejoras que deben introducirse en el mecanismo, desde un punto de vista funcional, son las siguientes:

- **La congestión debe detectarse en las entradas y no sólo en las salidas**. Esto evitaría las posibles detecciones incorrectas de la raíz de árboles de congestión, lo que garantizaría poder determinar siempre con precisión qué paquetes pertenecen realmente a flujos congestionados y cuáles no.
- **Las notificaciones de congestión sobre puntos más específicos que los asociados a *SAQs* ya asignadas deben aceptarse en lugar de rechazarse**. De este modo, el mecanismo se adaptaría dinámicamente al movimiento "*downstream*" de las raíces de árboles de congestión, reconociendo todas las *SAQs* asignadas a un árbol la nueva posición de la raíz.
- **La liberación de *SAQs* debe realizarse de forma distribuida**. Esto es, las *SAQs* deberían poder liberarse independientemente de si se encuentran o no en una hoja del árbol (es decir, esta liberación no debería realizarse necesariamente de forma ordenada), lo cual permitiría un uso más eficiente de los recursos.

La introducción de cada una de estas mejoras en el funcionamiento del mecanismo no es trivial, ya que esto implica alterar notablemente ciertos procedimientos fundamentales establecidos en la propuesta original, afectando incluso, en algunos casos, a las estructuras de las memorias de datos y de control. Además, estas modificaciones tienen repercusiones colaterales en otros aspectos del mecanismo, que a su vez necesitan ser adaptados al nuevo esquema. En consecuencia, es conveniente explicar, por separado y con detalle, cómo pueden implementarse las distintas mejoras mencionadas. A ello están dedicados los siguientes puntos.

5.2.1. Detección de congestión en las entradas

En principio, el detectar congestión en las entradas podría parecer una cuestión trivial: bastaría monitorizar, como en las salidas, el nivel de ocupación de las colas estándar y "disparar la alarma" de congestión en el momento en que dicho nivel superase cierto umbral. Efectivamente, este método permitiría detectar, con bastante probabilidad de certeza, la presencia de congestión en una cola estándar de un conmutador.

Ahora bien, es necesario tener en cuenta que un aspecto clave de *RECN* es el identificar con precisión la posición de la raíz de cada árbol de congestión detectado, de cara a que las *SAQs* almacenen exclusivamente paquetes dirigidos a las raíces de árboles. Puesto que, como se ha explicado anteriormente, la raíz real de un árbol de congestión siempre se sitúa en una salida de un conmutador (por la que existe

contención), el hecho de que se detecte congestión en una entrada no quiere decir en absoluto que dicha entrada sea la raíz del árbol. Es decir, que si una cola estándar en una entrada se congestiona es porque existe contención por una salida en dicho conmutador, y dicha salida es la raíz real del árbol detectado. Dicha salida, que podría en principio ser cualquiera, no puede identificarse directamente mediante el simple método de detección descrito, que por tanto no proporciona automáticamente la posición de la raíz del árbol. Nótese que esta incertidumbre no existe cuando la detección se produce en las salidas, porque en ese caso la propia salida congestionada sí puede ser (y así se considera) la raíz de un árbol.

En definitiva, no basta con monitorizar el nivel de ocupación de las colas estándar de las entradas de cada conmutador, sino que es necesario emplear algún método que, además, nos permita, en caso de detección, determinar qué salida del conmutador es la "culpable" del estado de congestión en dicha entrada.

En este sentido, existen varias posibles soluciones. Por ejemplo, una aproximación "probabilística" podría consistir en contar la cantidad de paquetes almacenados en la cola estándar congestionada que solicitan cada salida, de modo que la salida con un mayor número de paquetes solicitantes sería considerada la salida congestionada. Este planteamiento tiene el inconveniente de no garantizar con seguridad la identificación correcta de la salida responsable (esto es, la detección de la raíz del árbol), pero sobre todo resultaría muy dificultoso de implementar en un sistema real, debido a la necesidad de inspeccionar las cabeceras de todos los paquetes almacenados.

Otra solución, más sencilla de implementar, consistiría en considerar como raíz del árbol la salida solicitada por el paquete almacenado en cabeza de la cola estándar de la entrada congestionada. Pese a la sencillez de este planteamiento, es bastante probable que si este paquete está bloqueado en una entrada, la salida correspondiente sea un punto de congestión. Pero, desde luego, este método tampoco ofrece garantías plenas de identificar correctamente la raíz del árbol.

Frente a estos métodos "estadísticos", existe una solución que sí garantiza identificar sin dudas la salida responsable de la congestión en una entrada. Dicha solución consiste en dividir la cola estándar en tantas colas como puertos de salida existan en el conmutador, almacenando cada paquete entrante en la cola asociada a la salida solicitada por dicho paquete (es decir, se transformaría la cola estándar en varias colas siguiendo un esquema similar a *Virtual Output Queuing* a nivel de conmutador). Este esquema permite detectar congestión en las entradas del modo indicado (monitorizando la ocupación de cada una de estas colas), y al mismo tiempo identificar automáticamente a la salida responsable de la congestión, que sería la cola de salida asociada a la cola de entrada congestionada. Puesto que este sistema es preciso en cuanto a la detección de congestión, y puesto que la implementación real de estas colas no resultaría especialmente complicada, nos inclinamos por su uso.

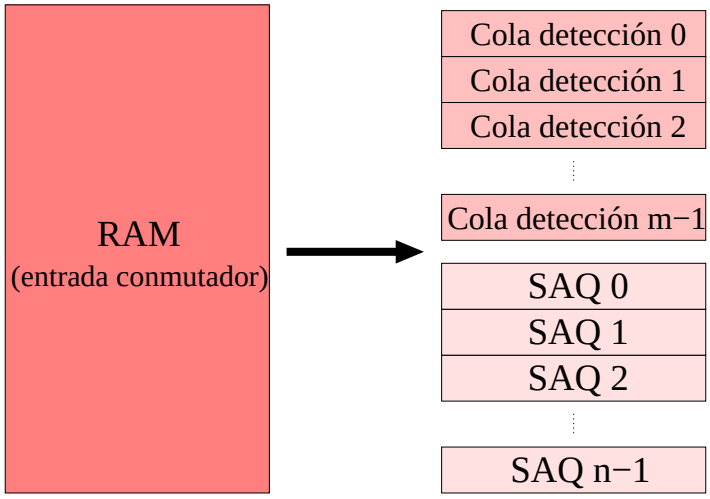


Figura 5.12: Distribución de la *RAM* de una entrada según la versión mejorada de *RECN*.

Así pues, en adelante asumiremos que la congestión en las entradas se detecta de este modo[4]. Además, para distinguir claramente a las colas resultantes de la división de la cola estándar nos referiremos a ellas en adelante con el nombre de **colas de detección**. Nótese que dichas colas sólo existirían en las entradas de los conmutadores (nunca en las salidas), y que además son independientes de las *SAQs* de dicha entrada. Es decir, la memoria de una entrada de un conmutador se compartiría entre las colas de detección y las *SAQs* activas en un instante dado en dicha entrada. La figura 5.12 muestra cómo debería organizarse la *RAM* de datos de las entradas de los conmutadores para implementar este método de detección de congestión en las mismas.

Sin embargo, y aunque mediante el uso de las colas de detección se resuelve el problema de la identificación de la raíz del árbol, queda pendiente un asunto de suma importancia. Cuando se detecta congestión en las salidas del conmutador (ver sección 4.3.1) no es necesario asignar *SAQs* en dicho punto porque, al ser dicho punto la raíz de un árbol, todos los paquetes que pasen por él pueden considerarse congestionados, no habiendo paquetes "fríos" que separar. Pero, de nuevo, es preciso no olvidar que en el caso de detectar congestión en las entradas, el punto de detección no es nunca la raíz de un árbol, y por tanto por dicha entrada pasarán paquetes dirigidos hacia la raíz del árbol (una salida concreta del mismo conmutador) y paquetes dirigidos a otras salidas. Esto lleva a la necesidad de separar ambos tipos de paquetes para evitar el *HOL blocking*. En consecuencia, cuando se detecte congestión en una entrada será necesario a la vez asignar inmediatamente en dicha entrada una *SAQ* para contener los paquetes dirigidos hacia la raíz del árbol.

[4]En cualquier caso, téngase en cuenta que el uso de este método concreto no resulta imprescindible para el funcionamiento del mecanismo global, y que podría emplearse cualquier otro sistema que permitiera obtener la información que éste proporciona.

Nótese que el punto de congestión asociado a dicha *SAQ* será una salida del mismo conmutador (la raíz del árbol identificada), y que la ruta (*turnpool*+máscara) almacenada en el registro *CAM* correspondiente consistirá únicamente en un desplazamiento (entre la entrada donde se encuentra la *SAQ* y la salida donde se sitúa la raíz del árbol). Por tanto, la *SAQ* está destinada a contener los flujos de paquetes que hasta entonces se almacenaban en una cola de detección, que precisamente no es otra que aquélla en la que se ha detectado congestión. Puesto que todos los paquetes contenidos en ese momento en la cola de detección deberían almacenarse en la *SAQ*, es conveniente transferirlos automáticamente a ésta. De cara a una implementación real esto puede implementarse sencillamente mediante un intercambio ("*swapping*") de los punteros que representan ambas colas, por lo que no existiría un movimiento "físico" de los datos. Además, nótese que la cola de detección no recibirá más paquetes mientras la *SAQ* esté activa, y por tanto puede inhabilitarse temporalmente (en principio, hasta que la *SAQ* se libere). A su vez, la *SAQ* puede activarse y emitir paquetes directamente.

Por otra parte, al pasar los paquetes "en bloque" a la *SAQ*, la ocupación de ésta se encontrará probablemente por encima del umbral de propagación de congestión (siempre que éste sea igual o menor que el umbral de detección). De ser así, la *SAQ* debería notificar automáticamente congestión al puerto inmediato en sentido *upstream*.

En la figura 5.13 se muestran todas las acciones implicadas en el proceso de detección de congestión en una entrada. Concretamente, la figura 5.13.a muestra la situación en el conmutador hasta justo el instante de la detección. En la figura se asume que el valor de *speedup* del conmutador es tal que la congestión se produce antes en las entradas. Puede apreciarse cómo la congestión se detecta al superar la ocupación de una cola de detección concreta (la asociada a la salida $P5$ en la entrada $P1$) el umbral de detección. Evidentemente, la raíz del árbol es automáticamente identificada como la salida asociada ($P5$) a la cola de detección. A su vez, la figura 5.13.b representa un instante inmediatamente posterior a la detección de congestión. Puede comprobarse cómo la detección ha provocado la asignación de una *SAQ* (*SAQ0*) en $P1$, que se asocia a la salida $P5$ del conmutador. Dicha *SAQ*, como puede apreciarse, recibe automáticamente todos los paquetes contenidos anteriormente en la cola de detección asociada a $P5$, la cual se inhabilita. También puede observarse en la figura el *turnpool* y la máscara contenidos en la notificación de congestión que se enviará en sentido *upstream* en cuanto se tranfieran los paquetes desde la cola de detección a *SAQ0*. Obviamente, la ruta indicada en esta notificación es la que lleva a la salida $P5$ desde la entrada $P1$ (un único desplazamiento).

Cabe preguntarse si este método de detección en las entradas soluciona realmente los problemas que son consecuencia de sólo detectar congestión en las salidas. Para comprobarlo, la figura 5.14 representa cómo afrontaría el nuevo mecanismo la situación de partida mostrada en el ejemplo de la figura 5.8. Como puede verse en la figura 5.14.a, la acumulación de paquetes de los flujos calientes en las colas de las entradas (colas de

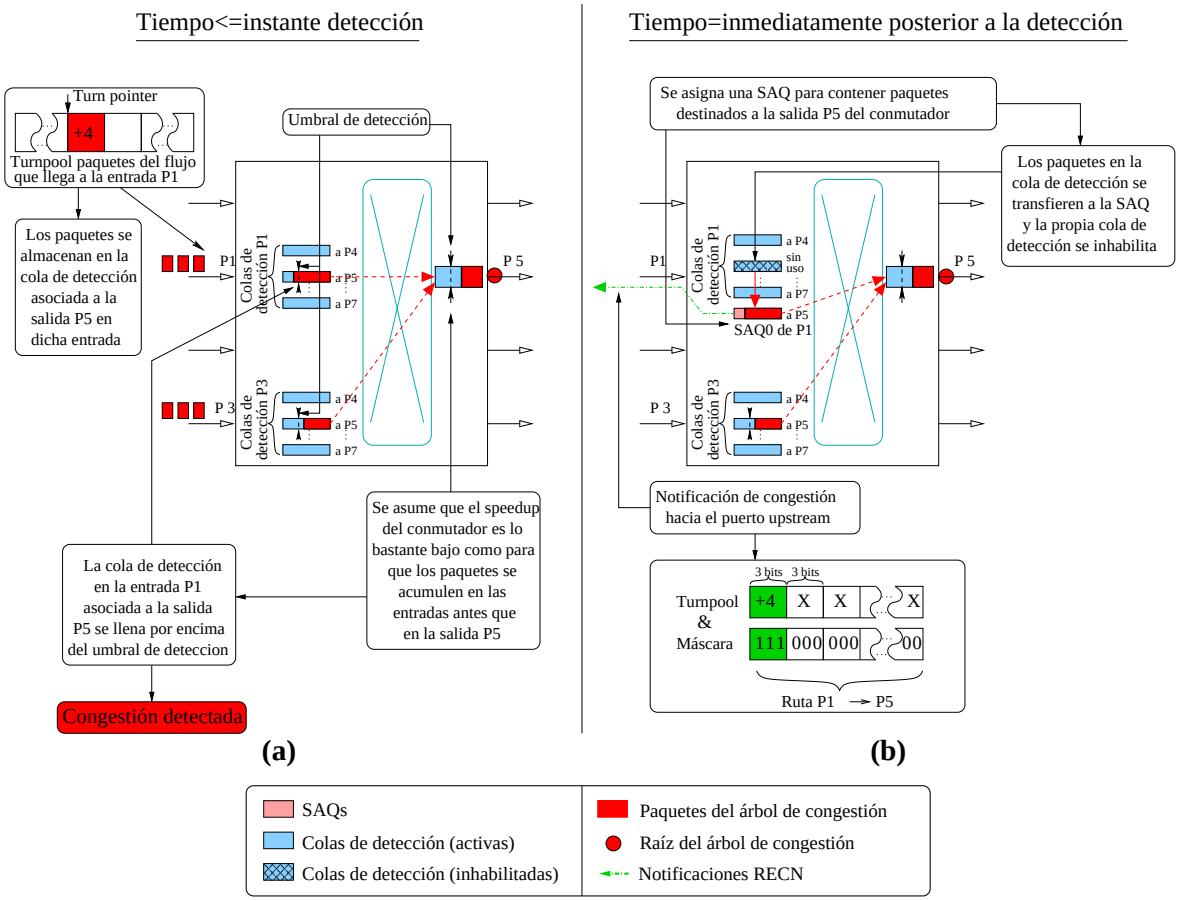


Figura 5.13: Detección de congestión en una entrada de un conmutador.

detección, en este caso) del conmutador 2 lleva ahora a que se asignen $SAQs$ en dichas entradas para contener paquetes dirigidos a la salida $P6$ (instante $t0$). Esto implica que, en lugar de activarse el control de flujo hacia la cola estándar de la salida $P5$ del conmutador 1, ahora se enviará inmediatamente una notificación de congestión hacia dicha salida. Esto provocará allí la asignación de una SAQ en la que se almacenarán desde entonces los paquetes del flujo congestionado, evitándose así la congestión de la cola estándar en este punto. La posterior llegada masiva de paquetes a esta SAQ provocaría a su vez (instante $t0 + T$) la notificación de congestión a la entrada $P1$ del conmutador 1, donde se asignaría consecuentemente una SAQ.

Nótese que, a diferencia de lo que sucede en el ejemplo de la figura 5.8, en esta ocasión todas las $SAQs$ asignadas en la figura 5.14 están asociadas al mismo punto de congestión (salida $P6$ del conmutador 2), donde se sitúa exactamente la raíz del árbol. Este hecho fundamental permite (figura 5.14.b) que, en caso de llegar un nuevo flujo ("frío") a la salida $P1$ del conmutador 1, los paquetes de dicho flujo sean separados totalmente de los paquetes congestionados, y no sólo en dicho punto, sino en todo su

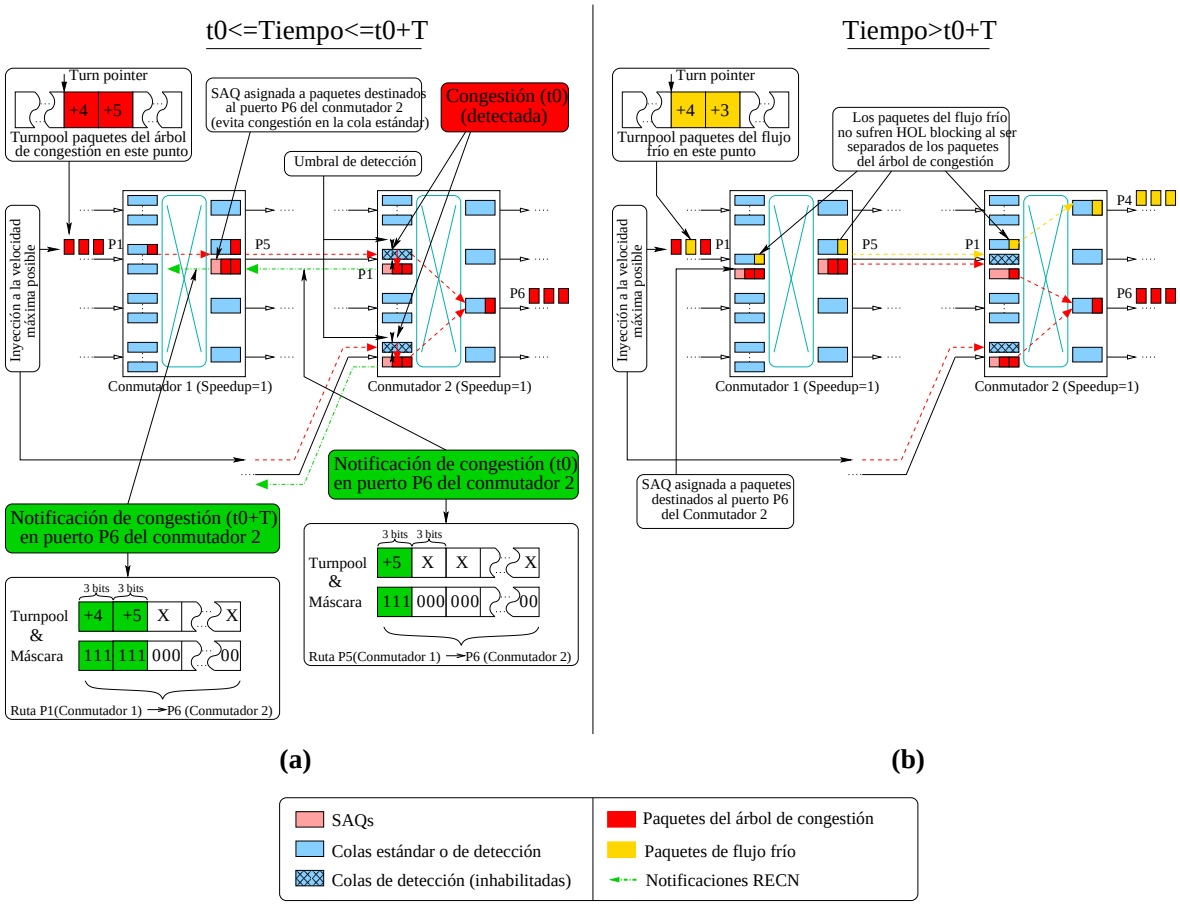


Figura 5.14: La identificación correcta de la raíz de un árbol permite la separación total de flujos fríos y calientes.

recorrido, como puede verse en la figura. En consecuencia, se evita el *HOL blocking* que una detección incorrecta de la raíz pudiera producir.

En definitiva, el método explicado en la presente sección permite detectar con exactitud el punto de aparición de la congestión, de cara a asignar correctamente los recursos destinados a evitar el *HOL blocking*. De este modo, el uso de este método, que se asume desde ahora, evita una de las causas de falta de eficacia del mecanismo original.

5.2.2. Aceptación de notificaciones de congestión más específicas

La razón de que la propuesta original de *RECN* establezca el rechazo de notificaciones más específicas no es otra que el garantizar la entrega en orden de todos los paquetes (ver sección 4.3.7.2). Conviene recordar que, concretamente, con este rechazo se impide que se asigne una *SAQ* asociada a cierta ruta X en un puerto donde existe

otra *SAQ* asociada a una ruta menos específica que X[5], pero que, por haber sido asignada anteriormente, puede contener paquetes que siguen la ruta X. Si la notificación se aceptara y se asignara la *SAQ*, los paquetes que llegaran al puerto a posteriori y siguieran la ruta X se almacenarían en esta *SAQ* más específica, tal y como está establecido para el caso de que la comparación de la ruta de un paquete entrante con las rutas asociadas a las *SAQs* activas en el puerto ofrezca varias coincidencias. Y esto llevaría a que dichos paquetes pudieran ser emitidos antes que los paquetes almacenados anteriormente en la *SAQ* asociada a la ruta menos específica y que se dirigen al mismo destino, desordenándose así el flujo de paquetes.

Por tanto, si deseamos que las notificaciones de congestión más específicas sean aceptadas, es necesario encontrar un método que garantice la entrega en orden de los paquetes en las circunstancias mencionadas. En este sentido, cabe recordar que la propuesta original establece un método (explicado también en la sección 4.3.7.2) para evitar la entrega fuera de orden de los paquetes almacenados en la cola estándar respecto de los almacenados en *SAQs*. Se trata de la introducción al final de la cola estándar de un *link pointer* hacia una *SAQs* recién asignada, que estará bloqueada (no podrá emitir paquetes) hasta que dicho *link pointer* llegue a la cabeza de la cola estándar. Recordemos también que este bloqueo se implementa mediante el bit de bloqueo del registro *CAM* correspondiente.

Evidentemente, este método también podría ser usado para evitar la entrega fuera de orden de los paquetes almacenados en una *SAQ* respecto de los almacenados en otra *SAQ*. De hecho, esta posibilidad se consideró durante el diseño de la propuesta original de *RECN*, pero se descartó debido a la (errónea) idea de que el rechazo de notificaciones más específicas no plantearía problemas, y también debido a que el uso de *link pointers* entre *SAQs* introduce ciertas complicaciones operativas en el mecanismo global. Obviamente, al vernos obligados a aceptar notificaciones más específicas para no perder eficacia, el uso de este método resulta imprescindible.

Las complicaciones mencionadas en el párrafo anterior son debidas a que, a diferencia del sistema de *link pointers* exclusivamente colocados en la cola estándar (que se pueden colocar automáticamente al final de la misma en cuanto se asigna una nueva *SAQ*) no es trivial decidir el lugar (la cola) donde debe colocarse un *link pointer* entre *SAQs*, o incluso si dicho *link pointer* es realmente necesario. Más concretamente, los *link pointers* entre *SAQs* no serán necesarios si no existen en el puerto que recibe la notificación de congestión *SAQs* activas asociadas a rutas menos específicas que la ruta notificada. Por tanto, es necesario comparar la ruta notificada con la asociada a todas las *SAQs* activas para comprobar si alguna de ellas es una subruta (con origen en el puerto) de la ruta notificada. Caso de no existir en el puerto ninguna *SAQ* activa

[5]Es esencial tener presente que al referirnos a rutas menos y más específicas indicamos siempre que la más específica es una ampliación de la menos específica, por lo que ambas rutas coinciden en todos los desplazamientos de los que conste la menos específica.

asociada a una ruta menos específica, el *link pointer* hacia la *SAQ* recién asignada debe colocarse, tal y como está establecido en la propuesta original, al final de la cola estándar (o, en el caso de las entradas, al final de la cola de detección asociada a la salida desde la que se recibe la notificación). Evidentemente, si la cola estándar o de detección estuviera vacía, la nueva *SAQ* no llegaría a bloquearse y podría emitir paquetes inmediatamente.

Al contrario, si se comprueba que sí existe en el puerto una *SAQ* activa asociada a una ruta menos específica que la notificada, debe colocarse al final de dicha *SAQ* el *link pointer* hacia la *SAQ* recién asignada. Al igual que en el caso de *link pointers* colocados en las colas estándar, la *SAQ* recién asignada estará bloqueada hasta que dicho *link pointer* llegue a la cabeza de la *SAQ* que lo contiene. Evidentemente, esto es necesario para garantizar la entrega en orden de los paquetes, puesto que en la *SAQ* asociada a la ruta menos específica puede haber almacenados paquetes que siguen la ruta asociada a la nueva *SAQ*.

Ahora bien, también puede suceder que existan en el puerto varias *SAQs* activas asociadas a rutas menos específicas que la ruta notificada: todas estas rutas serían distintas subrutas de la ruta notificada, cada una con una longitud diferente. En este caso, el problema es que en todas estas *SAQs* puede haber paquetes que siguen la ruta notificada, por lo que no es trivial decidir en cuál de ellas debe colocarse el *link pointer* a la nueva *SAQ*. Esta incertidumbre se resuelve estableciendo para estos casos la norma de colocar el *link pointer* a la nueva *SAQ* siempre en la *SAQ* asociada a la ruta menos específica de mayor longitud. Nótese que, caso de haber paquetes que siguen la ruta notificada en otra *SAQ* distinta a esta última, es porque ésta se ha asignado después de almacenarse dichos paquetes, y por tanto, según la propia norma establecida, existiría otro *link pointer* tras estos paquetes que bloquearía a su vez a la *SAQ* asociada a la ruta menos específica de mayor longitud, garantizándose así que los paquetes almacenados en primer lugar se emitirían antes que los almacenados a posteriori. Es decir: la norma implica un "bloqueo en cascada" entre varias *SAQs* en caso de ser necesario, de modo que una *SAQ* estará bloqueada hasta que salga el *link pointer* situado en otra *SAQ* asociada a una ruta menos específica que a su vez puede estar bloqueada hasta que salga otro *link pointer* situado en otra *SAQ* asociada a una ruta todavía menos específica.

En resumen, asumiremos que, en caso de recibirse en un puerto una notificación de congestión, se asignará una *SAQ* asociada a la ruta notificada y se colocará un *link pointer* para bloquear dicha *SAQ* en otra de las colas del puerto, según el siguiente criterio:

- Si no existen en el puerto *SAQs* asociadas a rutas menos específicas que la notificada, el *link pointer* se colocará al final de la cola estándar de dicho puerto (o

en la cola de detección correspondiente, caso de ser una entrada). Si dicha cola está vacía el *link pointer* es innecesario.

- Si existe en el puerto una sola *SAQ* asociada a una ruta menos específica que la notificada, el *link pointer* se colocará al final de dicha *SAQ*.
- Si existen en el puerto varias *SAQs*, asociadas cada una a una ruta distinta y menos específica que la notificada, el *link pointer* se colocará al final de la *SAQ* asociada a la ruta menos específica de mayor longitud.

Nótese que una *SAQ* puede estar asociada a una ruta que es menos específica que varias rutas al mismo tiempo (las más específicas serían distintas ampliaciones de la menos específica). Por tanto, es posible que en una misma *SAQ* existan simultáneamente varios *link pointers* hacia distintas *SAQs*, en el caso de que se reciban notificaciones de distintas rutas más específicas en un espacio breve de tiempo. Naturalmente, dichos *link pointers* serían totalmente independientes entre sí.

La figura 5.15 muestra un ejemplo del nuevo funcionamiento del mecanismo ante la recepción en un puerto de una notificación de congestión más específica. En la figura se asume que, antes de recibirse la notificación sobre el punto C en la entrada $P3$ del conmutador 1, las *SAQs SAQ0* y *SAQ1* de dicha entrada se encuentran activas y asociadas a los puntos A y B respectivamente. Además, se asume que en el instante de la recepción de la notificación la *SAQ1* se encuentra bloqueada por un *link pointer* colocado en *SAQ0*; por ello, el bit de bloqueo se encuentra activo en el registro *CAM*[6] de *SAQ1*. Como puede observarse en la figura, la mencionada notificación sobre el punto C es aceptada en la entrada, a pesar de corresponder a una ruta más específica que la que lleva a los puntos A y B. La aceptación de esta notificación supone la asignación en la entrada $P3$ del conmutador 1 de una *SAQ* (*SAQ2*), asociada al punto C (como puede comprobarse en el registro *CAM* correspondiente).

La figura 5.15 también indica que, siguiendo la norma explicada anteriormente, el *link pointer* a *SAQ2* debe colocarse en *SAQ1*, pues aunque tanto *SAQ0* como *SAQ1* están asociadas a rutas menos específicas que la notificada, la longitud de la ruta asociada a *SAQ1* es mayor. En la figura también se muestra el estado de las *SAQs* de la entrada $P3$ del conmutador 1, con los paquetes que podría almacenar cada una de ellas[7] tras la colocación del *link pointer* a *SAQ2*. Puede comprobarse que *SAQ0*, al estar asociada a la ruta menos específica de todas, podría contener paquetes dirigidos a los puntos A,B y C, pero sólo delante del *link pointer* a *SAQ1*; tras éste, sólo puede haber

[6] Nótese que en los registros *CAM* sólo se muestran los campos relevantes para el ejemplo.

[7] Conviene precisar que en esta parte de la figura cada paquete contiene el identificador del punto al que se dirige, lo que indica que el paquete pasará por este punto y por otros puntos de congestión más cercanos, pero no por puntos más lejanos. Así, los paquetes dirigidos al punto A pasan por éste, pero no por B ni por C; los paquetes dirigidos a B pasan por este punto y por A, pero no por C, y los paquetes dirigidos a C pasan por A, B y C.

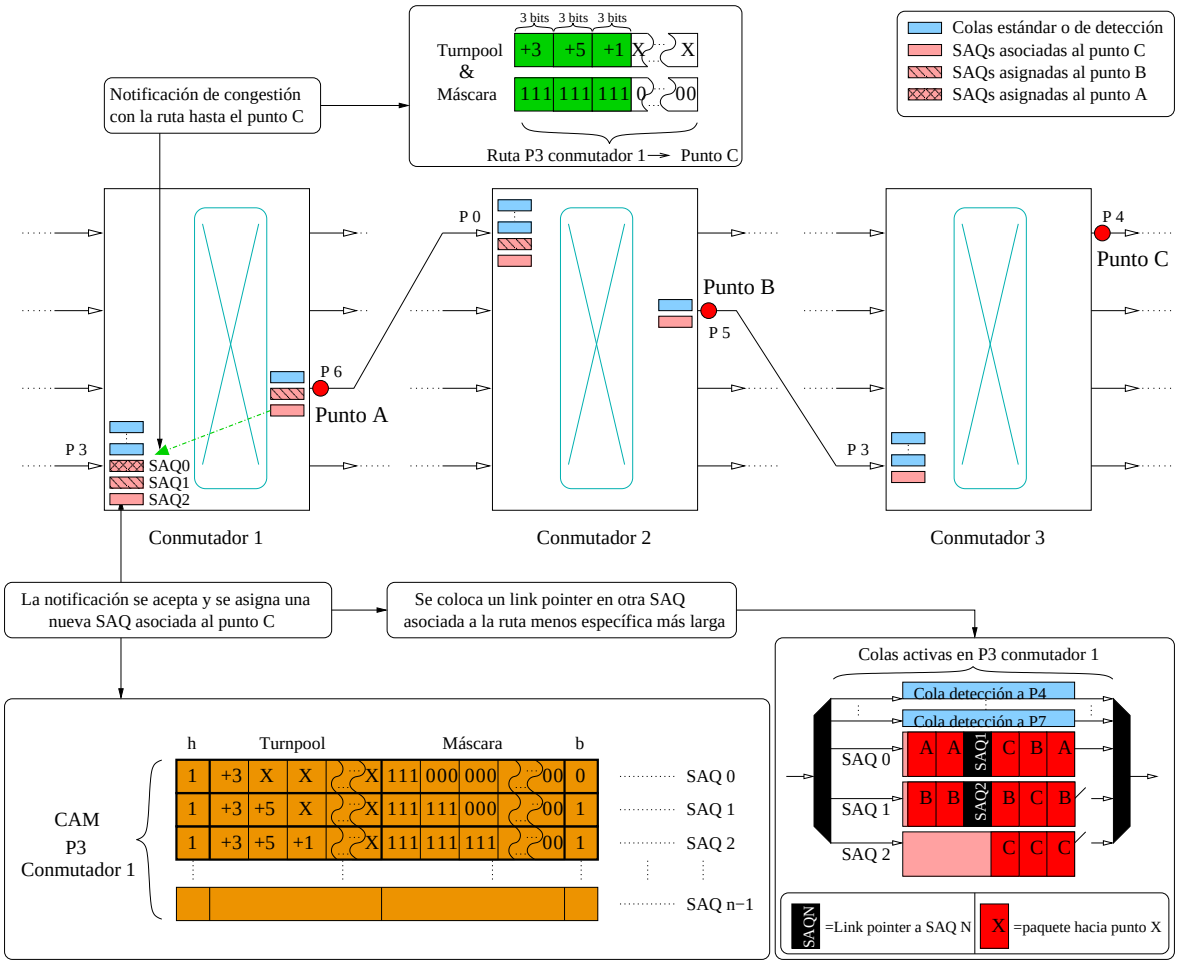


Figura 5.15: Asignación de *SAQs* a rutas más específicas y colocación de *link pointers* según el nuevo criterio.

paquetes dirigidos al punto A. Esto es así porque los paquetes para los puntos B y C se almacenarían en *SAQ1* tras activarse ésta (por ser más específica que *SAQ0*), o lo que es lo mismo, tras colocarse en *SAQ0* el *link pointer* a *SAQ1*. Análogamente, en *SAQ1*, tras el *link pointer* a *SAQ2*, sólo habrá paquetes dirigidos al punto B. Teniendo en cuenta que los distintos *link pointers* bloquean ordenadamente la emisión de paquetes desde las distintas *SAQs*, desde la menos a la más específica, todos los paquetes dirigidos a un punto concreto saldrán de $P3$ en el mismo orden de llegada.

Nótese que si en el ejemplo anterior llegaran al puerto $P3$ del conmutador 1 paquetes exclusivamente dirigidos al punto C, éstos se almacenarían en *SAQ0* sólo antes de asignarse *SAQ1*, y en ésta última únicamente hasta la asignación de *SAQ2*. En este caso, tanto *SAQ0* como *SAQ1* acabarían por vaciarse, lo que antes o después llevaría a su liberación. Es precisamente este efecto el que permite que, al aceptarse notificaciones más específicas, el mecanismo sí pueda asimilar ahora totalmente el desplazamiento de la posición de la raíz del árbol.

Para comprobarlo, veamos cómo resolvería ahora el mecanismo la misma situación que en la figura 5.9 fue analizada según el funcionamiento de la propuesta original. El diferente tratamiento de dicha situación a lo largo de varios instantes puede observarse en la figura 5.16. Recordemos que el problema a resolver se origina por el cambio de posición de la raíz del árbol en el instante $t0+T$, debido a la aparición de nuevos flujos, tras haberse detectado anteriormente ($t0$) una posición de la raíz situada en sentido "*upstream*" respecto a la nueva posición.

En los instantes iniciales (figuras 5.16.a y 5.16.b) el funcionamiento del mecanismo es prácticamente el mismo que en la propuesta original (con la única diferencia de que los paquetes que llegan a las entradas se almacenan en colas de detección en lugar de en colas estándar). Es decir: las diferentes posiciones de la raíz del árbol se detectan y se notifican de forma similar a la establecida en la propuesta original, produciendo las mismas asignaciones de *SAQs*.

Pero a partir del instante $t0+2T$ (figura 5.16.c) el mecanismo funcionaría de modo distinto. En ese instante, se recibirá en la entrada $P1$ del conmutador 1 una notificación sobre una ruta más específica (la que lleva desde ese punto a la salida $P6$ del conmutador 2) que la ruta asociada a la *SAQ* (*SAQ0*) activa en dicho puerto (ruta desde el puerto a $P5$ en el mismo conmutador). Puesto que ahora las notificaciones más específicas deben aceptarse, se asignará en la entrada $P1$ del conmutador 1 una *SAQ* (*SAQ1*) destinada a contener paquetes dirigidos a la salida $P6$ del conmutador 2 (se colocará un *link pointer* en *SAQ0* hacia *SAQ1*, pero esto carece de importancia para el ejemplo). De este modo, todos los paquetes del árbol de congestión que lleguen a esta entrada se almacenarán en *SAQ1*, mientras que *SAQ0* se irá vaciando. Por tanto, ante la llegada posterior a esta entrada (figura 5.16.d) de paquetes de un flujo frío que pase por $P5$ del conmutador 1 pero no por $P6$ del conmutador 2, no se producirá mezcla de estos paquetes con paquetes del árbol. Nótese que es posible que los paquetes del flujo frío se almacenen en *SAQ0* (tal y como se ha representado en la figura) si esta cola no ha podido liberarse todavía, o bien se almacenen en una cola de detección del puerto, si *SAQ0* se hubiera liberado antes de la llegada del flujo frío. Pero en ninguno de los dos casos se mezclarán estos paquetes con paquetes del árbol de congestión al almacenarse ahora estos últimos en *SAQ1*. En consecuencia, se evita el *HOL blocking* que la propuesta original permitía al mezclarse en aquel caso paquetes "fríos" y "calientes" ante la misma situación.

En conclusión, la aceptación de notificaciones más específicas no sólo es posible si se toman precauciones (el sistema de *link pointers* entre *SAQs*) para evitar la entrega fuera de orden de los paquetes, sino que además, y como se ha comprobado, es beneficiosa para el mecanismo, pues garantiza a éste reconocer y asimilar los desplazamientos "*downstream*" de las raíces de los árboles de congestión, lo cual es esencial para eliminar correctamente el *HOL blocking*. Lógicamente, en adelante asumiremos la aceptación de notificaciones más específicas, así como el uso de los correspondientes *link pointers* entre *SAQs*.

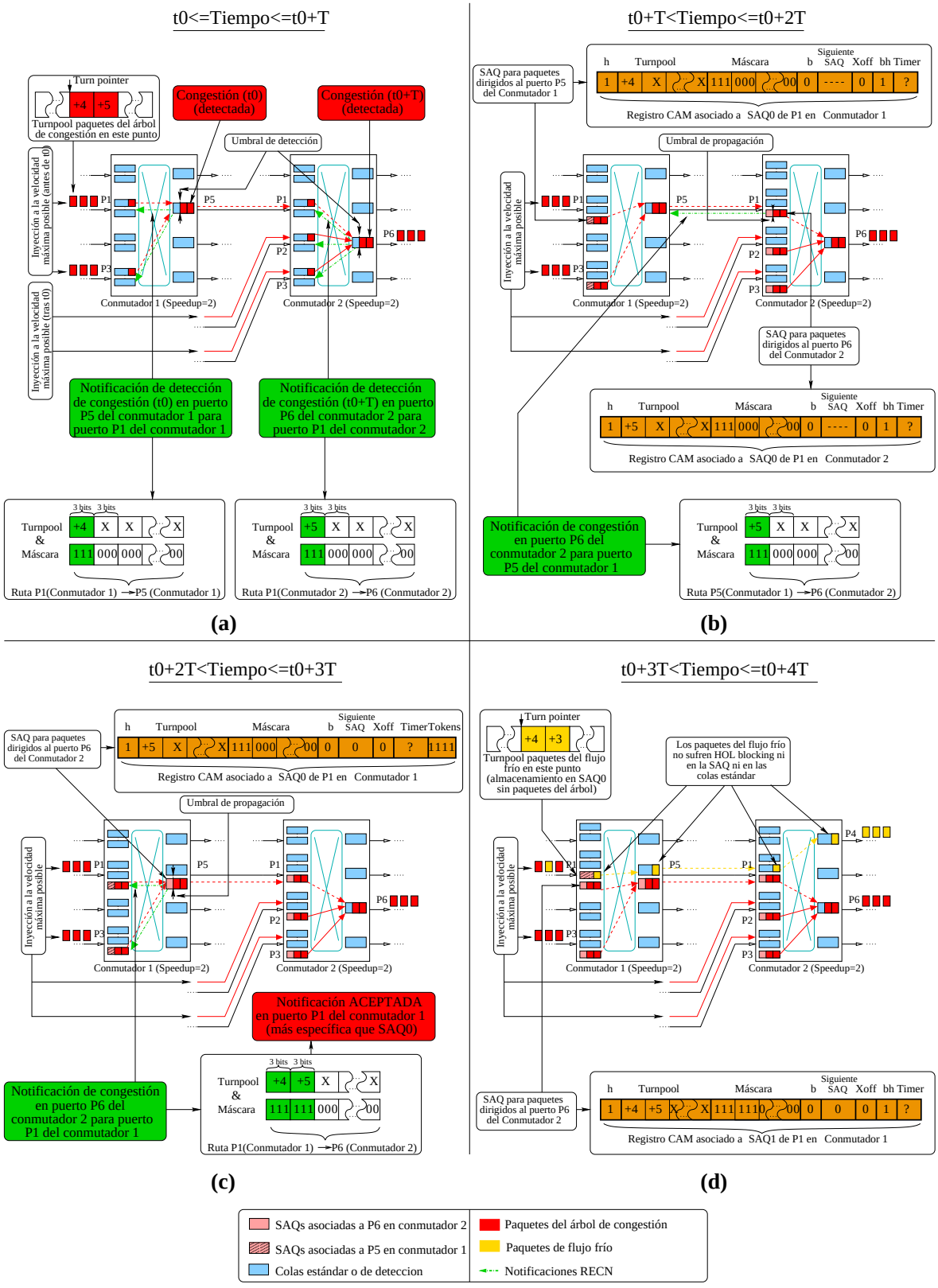


Figura 5.16: La aceptación de notificaciones más específicas permite asimilar el cambio de posición de la raíz del árbol de congestión.

5.2.3. Liberación de recursos distribuida

Como se ha explicado anteriormente, la liberación ordenada de *SAQs*, basada en la idea de una desaparición de los árboles invariablemente desde las hojas hasta la raíz, lleva a un desperdicio de recursos en aquellos casos en que los árboles no desaparezcan de este modo.

En consecuencia, es necesario proponer un nuevo sistema de liberación de *SAQs*, que contemple que los árboles pueden desaparecer de cualquier forma. En este nuevo sistema, por tanto, una *SAQ* debe poder liberarse independientemente de si se encuentra o no en una hoja del árbol de congestión al que está asignada o, lo que es lo mismo, independientemente de si existen o no *SAQs* activas en sentido "*upstream*" destinadas a contener paquetes del mismo árbol de congestión.

Ahora bien, al mismo tiempo, el nuevo sistema de liberación debe seguir garantizando que una *SAQ* situada en un punto de la red sólo se liberará cuando realmente no sea necesaria, es decir, cuando no exista posibilidad de que paquetes congestionados pertenecientes al árbol para el que está asignada la *SAQ* provoquen *HOL blocking* en dicho punto.

Teniendo en cuenta lo anteriormente expuesto, la decisión de liberar una *SAQ* asignada a cierto árbol de congestión debería basarse en detectar si dicho árbol sigue existiendo o no en el punto donde se encuentra la *SAQ* (o en un punto cercano), exclusivamente a partir de información disponible localmente. Esto permitiría efectivamente desligar la liberación de una *SAQ* del estado de cualquier otra *SAQ*, garantizando al mismo tiempo que esta liberación no es prematura.

Aunque la idea de evaluar la desaparición de un árbol de congestión en un punto a partir de información local implica asumir cierta probabilidad de error, es posible hallar condiciones que permitan realizar esta evaluación con casi total seguridad de acierto. En este sentido, nuestra propuesta concreta es permitir que una *SAQ* se libere únicamente cuando se cumplan todas las condiciones siguientes:

- **La *SAQ* está vacía**. Esta condición es obvia, puesto que si existen paquetes pertenecientes al árbol de congestión almacenados en la *SAQ*, el árbol sigue existiendo en el punto donde se encuentra la *SAQ*.
- **La *SAQ* no está bloqueada para emitir paquetes**. El bloqueo al que hace referencia esta condición es aquél que se implementa por medio del bit de bloqueo en los registros *CAM* y los *link pointers*, para evitar entrega de paquetes fuera de orden. Nótese que, caso de existir, un *link pointer* a una *SAQ* estará situado en otra cola que pudiera contener paquetes del árbol de congestión al que está asignada dicha *SAQ*. Puesto que una *SAQ* bloqueada implica que existe un *link pointer* hacia dicha *SAQ* en alguna otra cola, al impedir la liberación de una *SAQ*

en este estado se evita que esta liberación se produzca pudiendo haber paquetes pertenecientes al árbol de congestión en otras colas del mismo puerto (lo que indicaría que el árbol sigue presente en el puerto).

- **La *SAQ* no está en estado *Xoff***. Nótese que para que una *SAQ* se encuentre en estado *Xoff* debe mantenerla en este estado otra *SAQ* que estará asignada para el mismo árbol de congestión en un punto situado inmediatamente en sentido "*downstream*", y cuya ocupación será necesariamente media o alta. Teniendo esto en cuenta, esta condición garantiza que una *SAQ* no se liberará estando el árbol de congestión intensamente activo en las inmediaciones del punto donde se encuentra dicha *SAQ*.

Es especialmente interesante destacar que entre las nuevas condiciones para la liberación no se encuentra el *timer* que consideraba la propuesta original. Recordemos que dicho *timer* evitaba que una *SAQ* se liberara justo tras su asignación, y antes de recibir paquetes. Sin embargo, el *timer* ya no es necesario debido a que la segunda de las nuevas condiciones para la liberación cumple aproximadamente la misma función: aunque la *SAQ* estará vacía justo tras su asignación, no se liberará mientras pueda haber paquetes pertenecientes al árbol en otra cola del mismo puerto.

Nótese que, si es imposible que haya algún paquete del árbol en el puerto en el momento de la asignación de la *SAQ*, el uso de la segunda nueva condición permitiría la liberación inmediata de dicha *SAQ*, mientras que el *timer* no lo permitiría hasta pasado cierto tiempo. Pero, evidentemente, en estos casos (muy raros, por otra parte), la liberación de la *SAQ* es adecuada puesto que el árbol de congestión, con toda seguridad, no está presente en el puerto. En consecuencia, la nueva condición no sólo hace innecesario el uso del *timer*, sino que además permite al mecanismo reaccionar más rápidamente ante asignaciones incorrectas de *SAQs*. Más aún, el campo *timer* de los registros *CAM* puede eliminarse (junto con la lógica asociada) al no hacer falta para ninguna otra tarea del mecanismo, con el consiguiente ahorro de memoria *CAM*.

Igualmente, en las nuevas condiciones no se incluye el que una *SAQ* deba ser una hoja del árbol para liberarse, a diferencia de la propuesta original. Esto convierte en innecesario el bit de hoja (*SAQs* en entradas) y la lista de *tokens* (*SAQs* en salidas) de los registros *CAM* asociados a las *SAQs*, por lo que dichos bits también podrían eliminarse de la *CAM*.

Por otra parte, nótese que, puesto que ahora cualquier *SAQ* puede liberarse sin esperar una notificación de liberación de otra *SAQ* asignada para el mismo árbol y situada inmediatamente en sentido "*upstream*", ninguna *SAQ* puede tener garantías de que exista otra *SAQ* asignada para el mismo árbol y situada inmediatamente en sentido "*downstream*". Dicho de otro modo: puesto que cualquier *SAQ* puede liberarse independientemente del resto, el mecanismo no puede asumir que exista una "continuidad" en las *SAQs* asignadas para un árbol de congestión a lo largo de una de sus

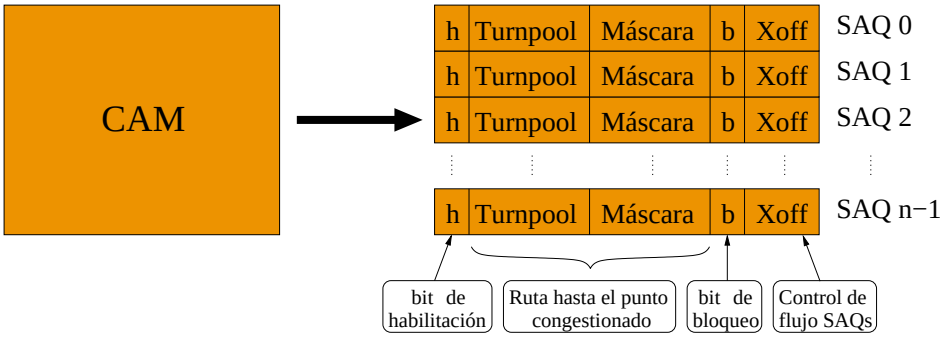


Figura 5.17: Nueva estructura de los registros de la *CAM*.

ramas. De este modo, el campo "Siguiente *SAQ*" de los registros *CAM* deja de tener sentido, y, al igual que en casos anteriores, puede eliminarse de los registros *CAM*.

En definitiva, las nuevas condiciones para la liberación de *SAQs*, además de permitir que ésta se realice de forma distribuida, permiten simplificar la estructura de la *CAM*, ya que la evaluación de dichas condiciones no requiere la información contenida en varios campos de los registros *CAM* que eran imprescindibles según la propuesta original de *RECN*. Concretamente, la figura 5.17 muestra la estructura simplificada de la *CAM*, ajustada a las necesidades reales del mecanismo tras las modificaciones introducidas.

Ahora bien, hay que tener en cuenta que la información contenida en el campo "Siguiente *SAQ*" era usada por el control de flujo *Xon/Xoff* entre *SAQs*. Concretamente, un identificador de la *SAQ* emisora se incluía en cada mensaje de este control de flujo, y en el puerto receptor dicho identificador era comparado con los almacenados en el campo "Siguiente *SAQ*" de todos los registros *CAM* asociados a *SAQs* activas en dicho puerto para determinar la *SAQ* receptora. Evidentemente, al no poder contar con la información del campo "Siguiente *SAQ*", el formato de los mensajes de control de flujo *Xon/Xoff* debe modificarse, sustituyendo el identificador de la *SAQ* emisora por otra información que permita determinar exactamente la *SAQ* receptora del mensaje. Una posibilidad es incluir en los mensajes de control de flujo el *turnpool* y la máscara asociados a la *SAQ* emisora, de modo que la *SAQ* receptora se determinaría comparando esta información recibida con el *turnpool* y la máscara almacenados en cada uno de los registros *CAM* asociados a *SAQs* activas en dicho puerto.

Otro aspecto interesante del nuevo mecanismo de liberación de *SAQs* es que no requiere el uso mensajes de control para funcionar. Cabe recordar que, según la propuesta original, la liberación de *SAQs* implicaba el intercambio de mensajes de control para notificar la liberación de una *SAQ* "*upstream*" a otra *SAQ* inmediata en sentido "*downstream*". Esto supone una gran ventaja, pues simplifica el funcionamiento del mecanismo y, quizá más importante, reduce el número de tipos de mensajes que el mecanismo maneja.

5.3. Evaluación de la propuesta mejorada

Una vez justificados y detallados los cambios introducidos en *RECN*, el siguiente paso es, lógicamente, realizar la pertinente evaluación para constatar si dichos cambios efectivamente mejoran el mecanismo. Téngase en cuenta que esta mejora sólo será relevante si permite a *RECN* alcanzar unas prestaciones acordes con los objetivos iniciales planteados en el presente estudio. En definitiva, se trata de comprobar si la propuesta mejorada de *RECN* es ahora capaz de eliminar el *HOL blocking* de forma realmente eficiente y escalable, e independientemente de la carga de tráfico o de las características de la red.

Al igual que en la parte análoga del capítulo anterior, los siguientes puntos están dedicados tanto a explicar el procedimiento seguido para realizar la nueva evaluación como a presentar los resultados obtenidos en la misma.

5.3.1. Método de evaluación

Teniendo en cuenta que las prestaciones que se desean obtener de la versión mejorada de *RECN* son idénticas a las que se deseaban obtener de la versión inicial, resulta evidente que la evaluación de la nueva versión debería realizarse siguiendo la misma metodología empleada en la evaluación de la propuesta inicial. En consecuencia, la evaluación de la nueva propuesta también se dividirá en una serie de análisis complementarios que permitirán en conjunto comprobar la validez del mecanismo como técnica eficiente y escalable de eliminación del *HOL blocking*.

Dichos análisis serán, en su mayoría, del mismo tipo que los expuestos en la evaluación precedente: ajuste de parámetros críticos, comparación de prestaciones del mecanismo con las obtenidas por otras técnicas de eliminación del *HOL blocking*, y análisis de la escalabilidad. Sin embargo, podemos adelantar que el resultado de estos análisis es mucho más positivo que en el caso anterior, y por ello merece la pena incluir en la presente evaluación algún otro análisis adicional. Estos nuevos análisis estarán orientados a analizar ciertas cuestiones que, aun siendo secundarias en cuanto a las prestaciones ofrecidas por el mecanismo, se plantearían en el caso de un hipotética implementación del mismo. Concretamente, se han añadido análisis sobre la sobrecarga de mensajes de control introducidos por el mecanismo y sobre el área de silicio requerida aproximadamente en cada puerto para la implementación de las memorias de datos que el mecanismo necesita.

En resumen, la evaluación de la versión mejorada de *RECN* se dividirá en los siguientes análisis[8]:

[8]En caso de duda respecto a la intención de los tres primeros, consúltese el capítulo anterior(sección 4.4.1).

1. **Ajuste de los parámetros críticos**.
2. **Análisis comparativo de prestaciones**.
3. **Análisis de escalabilidad**.
4. **Análisis de la sobrecarga de los mensajes de control**. Se trata de determinar si los paquetes de control empleados por *RECN* en su funcionamiento suponen una sobrecarga inaceptable para la red.
5. **Estimación de los requisitos de área de memoria**. Puesto que *RECN* requiere la existencia de ciertas estructuras de memoria de datos en los puertos, es conveniente conocer el área mínima necesaria para la implementación de dichas estructuras.

En cualquier caso, conviene precisar que los dos nuevos análisis se abordan de forma mucho menos exhaustiva (aunque no menos correcta) que los tres principales.

Por lo demás, en la presente evaluación se sigue usando el mismo método de obtención de resultados (simulación), y los mismos tipos de carga de tráfico (tráfico sintético y trazas) empleados en la evaluación anterior. Respecto a las razones del uso de estas opciones, y respecto a las características de las mismas, nos remitimos de nuevo al capítulo precedente (sección 4.4.1).

5.3.2. Herramienta de simulación

El simulador empleado para obtener los resultados necesarios para la nueva evaluación es el mismo empleado en la evaluación anterior (ampliamente descrito en la sección 4.4.2), con la lógica introducción de los cambios pertinentes para reflejar las mejoras de *RECN* propuestas en el presente capítulo.

Así pues, en lo que respecta al simulador, siguen siendo válidos todos los detalles indicados en la sección 4.4.2 en cuanto a modelado de la carga de tráfico, modelado de la red de interconexión, métricas ofrecidas y modelado de las distintas técnicas de control de congestión, con la excepción en este último punto de los detalles referentes al modelado de *RECN*.

En este sentido, los principales cambios en el simulador afectan a las estructuras de la memoria de los puertos, para ajustarlas al esquema establecido en la nueva versión de *RECN*. De este modo, la *RAM* de datos de cada entrada de un conmutador se modela ahora como compartida dinámicamente por todas las colas de detección y por todas las *SAQs* activas en la entrada en un momento dado. Nótese que en las salidas no existen colas de detección, por lo que el modelado de la memoria de datos en las salidas no varía respecto a la anterior versión del simulador. En cuanto a la memoria

de control (*CAM*) empleada para gestionar las *SAQs*, se ha modelado en entradas y salidas según las nuevas necesidades del mecanismo, desapareciendo de su estructura los campos indicados anteriormente (bit de hoja, *timer*, vector de *tokens*, Siguiente *SAQ*).

Obviamente, los cambios mencionados en el modelado de la memoria de datos permitieron a su vez modelar el nuevo sistema de detección en las entradas. El mismo parámetro del simulador que establece el umbral de detección es empleado ahora en la comprobación del nivel de ocupación de las colas "frías" tanto de la entradas (colas de detección) como de las salidas (cola estándar). También se ha modificado el simulador para que en los puertos se acepten notificaciones de congestión más específicas, con el modelado del correspondiente sistema de *link pointers* entre *SAQs* para garantizar la entrega en orden de los paquetes. Igualmente, se han cambiado en el simulador las condiciones requeridas para la liberación de una *SAQ*, de cara a modelar el nuevo sistema de liberación distribuida.

Del mismo modo, se ha variado el modelado de los mensajes de control empleados por *RECN*, suprimiendo directamente las ahora innecesarias notificaciones de liberación de *SAQ* y cambiando la información contenida en los paquetes de control de flujo *Xon/Xoff*, que ahora contienen el *turnpool* y la máscara asociados a la *SAQ* emisora.

Como es lógico, el simulador fue convenientemente sometido a pruebas exhaustivas para comprobar que su funcionamiento era el correcto tras la introducción de los cambios mencionados.

5.3.3. Ajuste de los parámetros críticos del mecanismo

Aunque podría considerarse la posibilidad de extrapolar las conclusiones obtenidas en el ajuste de parámetros críticos de la propuesta inicial (sección 4.4.3), el calibre de los cambios introducidos en *RECN* hace aconsejable realizar un nuevo ajuste para la propuesta mejorada. Esto es especialmente cierto si tenemos en cuenta que uno de los parámetros críticos del mecanismo, concretamente el nivel del umbral de detección, es clave en el nuevo sistema de detección de congestión en las entradas.

En consecuencia, en esta sección se presenta un nuevo ajuste de los parámetros críticos de *RECN*, considerando como tales, de nuevo, tanto los niveles de los umbrales de detección y control de flujo *Xon/Xoff* como el número de *SAQs* por grupo[9]. Evidentemente, este análisis tiene como objetivo determinar con qué valores (aproximados) de estos parámetros críticos obtiene la versión mejorada de *RECN* las mejores prestaciones.

[9]Los motivos por los que dichos parámetros se consideran críticos, así como las relaciones entre ellos, ya se explicaron en la sección 4.4.3. Además, nótese que los cambios introducidos para mejorar *RECN* no añaden nuevos parámetros críticos al mecanismo

		Tráfico uniforme			Tráfico árbol de congestión				
Caso de tráfico	Tamaño de red (BMIN)	Nº. de fuentes	Destino	Tasa de generación	Nº. de fuentes	Destino	Tasa de generación	Instante inicio	Instante fin
1	64×64	87,5 %	aleatorio	50 %	12,5 %	Terminal 32	100 %	800 μs	1100 μs
2	64×64	87,5 %	aleatorio	100 %	12,5 %	Terminal 32	100 %	800 μs	1100 μs
3	64×64	75 %	aleatorio	50 %	25 %	Terminal 32	100 %	800 μs	1100 μs
4	64×64	75 %	aleatorio	100 %	25 %	Terminal 32	100 %	800 μs	1100 μs

Tabla 5.1: Tráficos empleados en el ajuste de parámetros para la versión mejorada de *RECN*.

5.3.3.1. Configuración de las pruebas

Análogamente al anterior ajuste de parámetros, el presente análisis requiere la realización de una serie de simulaciones, usando en todas ellas la nueva versión de *RECN* como técnica de control de congestión, y empleando distintos valores de los parámetros críticos del mecanismo.

Concretamente, los distintos valores de los parámetros que se han considerado en este ajuste son los mismos que sirvieron para realizar el ajuste de la propuesta inicial. Es decir, en cuanto al número de *SAQ* por grupo, se han realizado simulaciones con 4, 8 y 16 *SAQs*. Y en cuanto a los niveles de los umbrales de detección, *Xon* y *Xoff* se han empleado las tres diferentes ternas de valores que pueden verse en la tabla 4.1 expresados en bytes[10]. Cabe recordar que estas tres ternas corresponden respectivamente a niveles altos, medios y bajos de estos umbrales.

En cuanto a la carga de tráfico, se han empleado los patrones de tráfico sintético que pueden verse en la tabla 5.1. Nótese que los casos de tráfico 1 a 4 son exactamente los mismos que los casos 1 a 4 empleados en la evaluación de la propuesta inicial. Recordemos que en estos patrones varía tanto la tasa de generación de tráfico uniforme como el porcentaje de fuentes que generan tráfico uniforme o congestionado. Sin embargo, no se ha considerado necesario realizar pruebas con patrones que supongan una duración distinta del intervalo de inyección de paquetes congestionados (caso de los patrones 5 a 8 empleados en el ajuste anterior), dadas las escasas (cuando no nulas) diferencias que presentaban los resultados obtenidos con estos patrones respecto al resto de resultados en la evaluación anterior.

Como ya se indica en la tabla anterior, y al igual que en el ajuste anterior, en lo que respecta a la estructura de la red sólo se ha considerado en este ajuste una red multietapa bidireccional (*BMIN*, patrón *perfect shuffle*) con 64 terminales. Las características de los conmutadores y enlaces de esta red son las mismas que se asumieron a lo largo de toda la evaluación precedente: conmutadores de 8 puertos (conectados bien a 4 terminales y 4 conmutadores, bien a otros 8 conmutadores), ancho de banda efectivo de los enlaces de 8 *Gbps* y memorias de 32 KB en cada entrada o salida. Ahora

[10] Respecto a los motivos del uso de estos valores concretos, nos remitimos a la sección 4.4.3.1.

bien, teniendo en cuenta la importancia que tiene el valor de *speedup* del *crossbar* de los conmutadores para el funcionamiento del mecanismo (según se comprobó en la evaluación de la propuesta inicial), el número de simulaciones realizadas para este análisis se ha duplicado para considerar dos valores distintos de *speedup*: 1,5 y 1.

Respecto a los *Input Adapters*, se sigue asumiendo que disponen de 32 KB de memoria en sus salidas, y también que dividen los mensajes generados en paquetes de 64 bytes previamente a su inyección en la red.

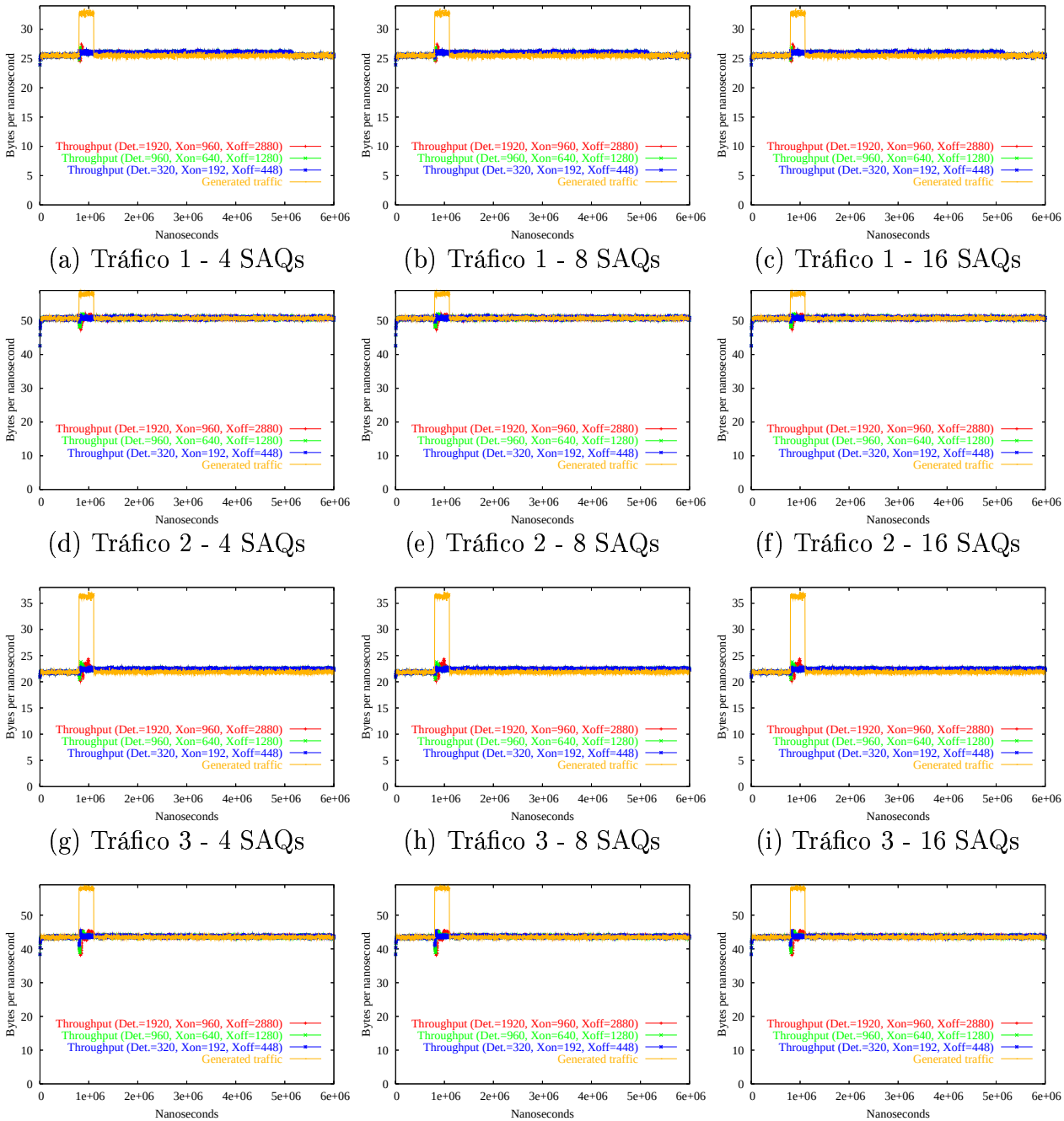
Los resultados que se muestran en la siguiente sección se han obtenido para todas las combinaciones posibles de parámetros críticos, patrones de tráfico y valores de *speedup* indicados. Estos resultados se corresponden a las métricas habituales para comprobar tanto la eficacia de *RECN* en la eliminación del *HOL blocking* (productividad de la red) como la cantidad de recursos necesaria para dicha eliminación (número de *SAQs* activas, tanto el número máximo en entradas o salidas como el número total en la red). En función de estos resultados, se decidirá qué configuración de parámetros críticos es más conveniente para el funcionamiento del mecanismo.

5.3.3.2. Resultados

Las figuras 5.18 y 5.19 muestran los resultados de productividad de la red en función del tiempo para todas las simulaciones realizadas de cara a este ajuste de parámetros. Más concretamente, la figura 5.18 muestra los resultados de productividad para todas las pruebas en las que se ha asumido un valor de *speedup* de 1,5, mientras que la figura 5.19 corresponde a aquellas pruebas realizadas para un valor de *speedup* de 1. Cada gráfica individual, a su vez, muestra varias series de resultados obtenidos para un caso de tráfico concreto y un único valor del número de *SAQs* por grupo. Cada una de estas series de resultados se corresponde a una terna de valores de los umbrales, salvo una de las series (en amarillo) que corresponde al tráfico total generado en la red a lo largo de la simulación.

Teniendo en cuenta los resultados mostrados en estas figuras, la primera impresión respecto a la nueva versión de *RECN* difícilmente puede ser mejor. Absolutamente en todos los casos (para cualquier terna de valores de umbrales, número de *SAQs*, trafico y valor de *speedup*) los resultados de productividad siguen el nivel marcado por el tráfico inyectado uniformemente (o sea, el nivel anterior y posterior al intervalo de inyección de tráfico congestionado) durante todo el tiempo de simulación. Esto significa que, en cualquiera de las configuraciones simuladas, la nueva versión de *RECN* mantiene la productividad al máximo posible incluso en presencia de tráfico congestionado.

De hecho, los resultados son tan positivos que es difícil decidir qué valores de los parámetros críticos son los más adecuados. Sólo se aprecian mínimos altibajos de los



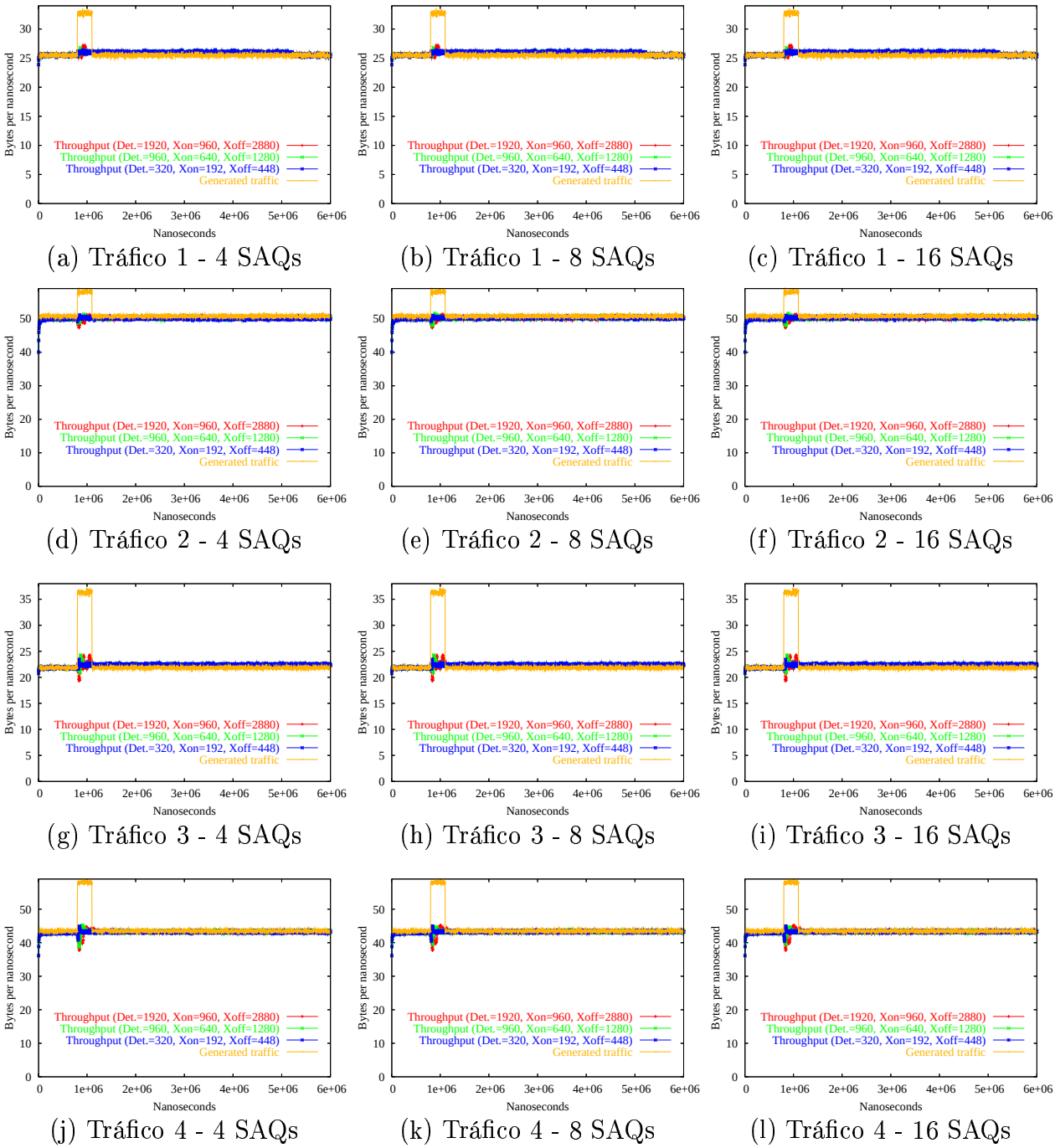
(a) Tráfico 1 - 4 SAQs (b) Tráfico 1 - 8 SAQs (c) Tráfico 1 - 16 SAQs

(d) Tráfico 2 - 4 SAQs (e) Tráfico 2 - 8 SAQs (f) Tráfico 2 - 16 SAQs

(g) Tráfico 3 - 4 SAQs (h) Tráfico 3 - 8 SAQs (i) Tráfico 3 - 16 SAQs

(j) Tráfico 4 - 4 SAQs (k) Tráfico 4 - 8 SAQs (l) Tráfico 4 - 16 SAQs

Figura 5.18: Productividad de la red en función del tiempo para las distintas configuraciones de parámetros *RECN* consideradas, y para los casos de tráfico 1 a 4, considerando conmutadores con *speedup*=1,5.



(a) Tráfico 1 - 4 SAQs (b) Tráfico 1 - 8 SAQs (c) Tráfico 1 - 16 SAQs

(d) Tráfico 2 - 4 SAQs (e) Tráfico 2 - 8 SAQs (f) Tráfico 2 - 16 SAQs

(g) Tráfico 3 - 4 SAQs (h) Tráfico 3 - 8 SAQs (i) Tráfico 3 - 16 SAQs

(j) Tráfico 4 - 4 SAQs (k) Tráfico 4 - 8 SAQs (l) Tráfico 4 - 16 SAQs

Figura 5.19: Productividad de la red en función del tiempo para las distintas configuraciones de parámetros *RECN* consideradas, y para los casos de tráfico 1 a 4, considerando conmutadores con *speedup*=1.

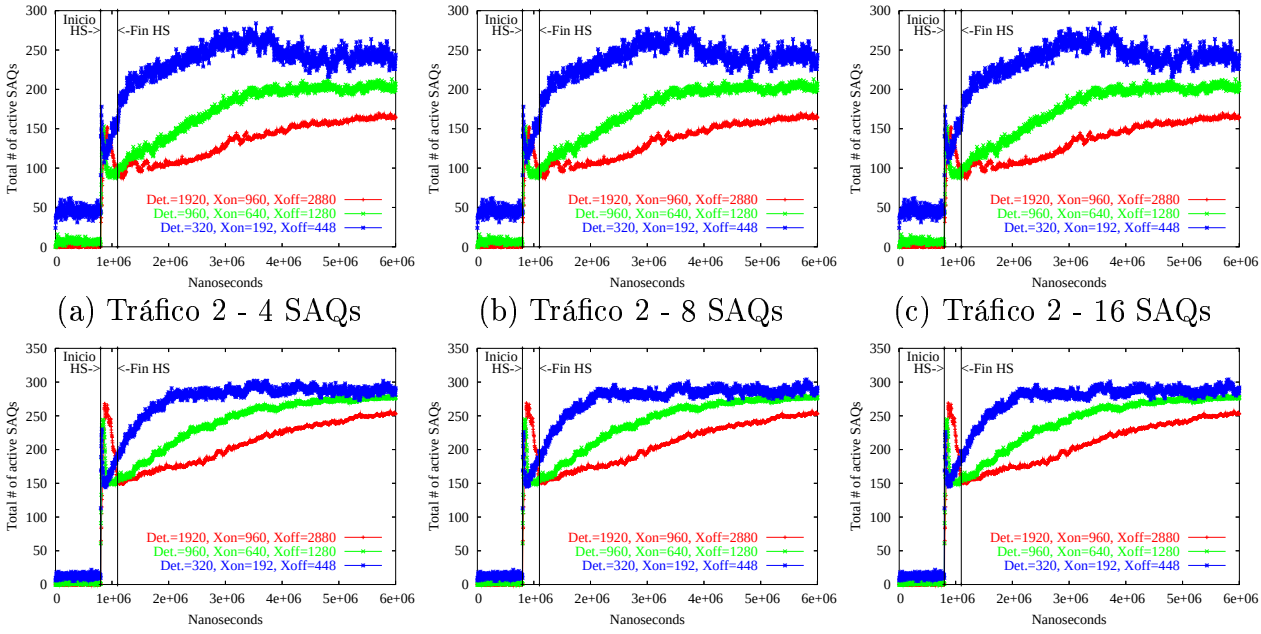
resultados de productividad justo en el comienzo de la inyección de tráfico congestionado, y sólo durante un periodo muy breve de tiempo. El máximo descenso de la productividad en estos altibajos es aproximadamente de un 10 %, en todas las gráficas correspondiente al tráfico 4 y sólo para la serie de valores altos de los umbrales. Los resultados de esta serie, además, son los que presentan una mayor inestabilidad (aunque mínima) en todos los casos. Aunque sólo sea por esta pequeña inestabilidad en la productividad obtenida, se refuerza la idea de que los valores altos de los umbrales suponen una reacción más lenta en el mecanismo, que acaba afectando a la productividad.

Respecto a las otras dos ternas de valores, no puede concluirse nada a partir de los resultados mostrados para ambos casos, pues éstos son demasiado parecidos. En cuanto al número de *SAQs*, parece que los resultados son independientes de dicho número, por lo que, en principio, puede pensarse que 4 *SAQs* por grupo serían suficientes para eliminar el *HOL blocking* en la red. Sin embargo, cualquier decisión en este sentido debe tomarse con toda la seguridad de que en ningún caso el mecanismo funcione incorrectamente por falta de *SAQs*

En cualquier caso, disponemos de otras métricas que pueden arrojar más luz sobre estas cuestiones. Concretamente, las figuras 5.20 y 5.21 muestran resultados sobre el número de *SAQs* activas en toda la red durante el tiempo de simulación. La figura 5.20 muestra resultados para redes con conmutadores de valor de *speedup*=1,5, mientras que la figura 5.21 muestra resultados para conmutadores con *speedup*=1. En ambos casos, y en beneficio de una mayor claridad, sólo se muestran resultados para los tráficos más intensos (tráficos 2 y 4). En cada figura individual se representan, para un tráfico y número de *SAQs* por grupo concretos, tres series con los resultados obtenidos usando las tres distintas ternas de valores de umbrales. Cada figura individual incluye además dos líneas como referencia de los momentos inicial y final, respectivamente, de la inyección de tráfico congestionado.

A la vista de estos resultados, se confirma la idea de que la terna de valores bajos de los umbrales implica un mayor consumo de *SAQs*, debido, evidentemente, a que en este caso el nivel de ocupación de las colas supera mucho más fácilmente el nivel de detección. Este hecho, como ya se ha indicado, puede llevar a que se detecten como situaciones de congestión aquéllas que no lo son realmente. Esto puede comprobarse al observar el número total de *SAQs* activas antes de la inyección de tráfico congestionado, que es mucho mayor para la terna de valores de umbrales bajos (especialmente para resultados obtenidos con el caso de tráfico 2, donde el porcentaje de tráfico uniforme es mayor). Por tanto, parece que sigue teniendo sentido seleccionar la terna de valores de umbrales medios [11].

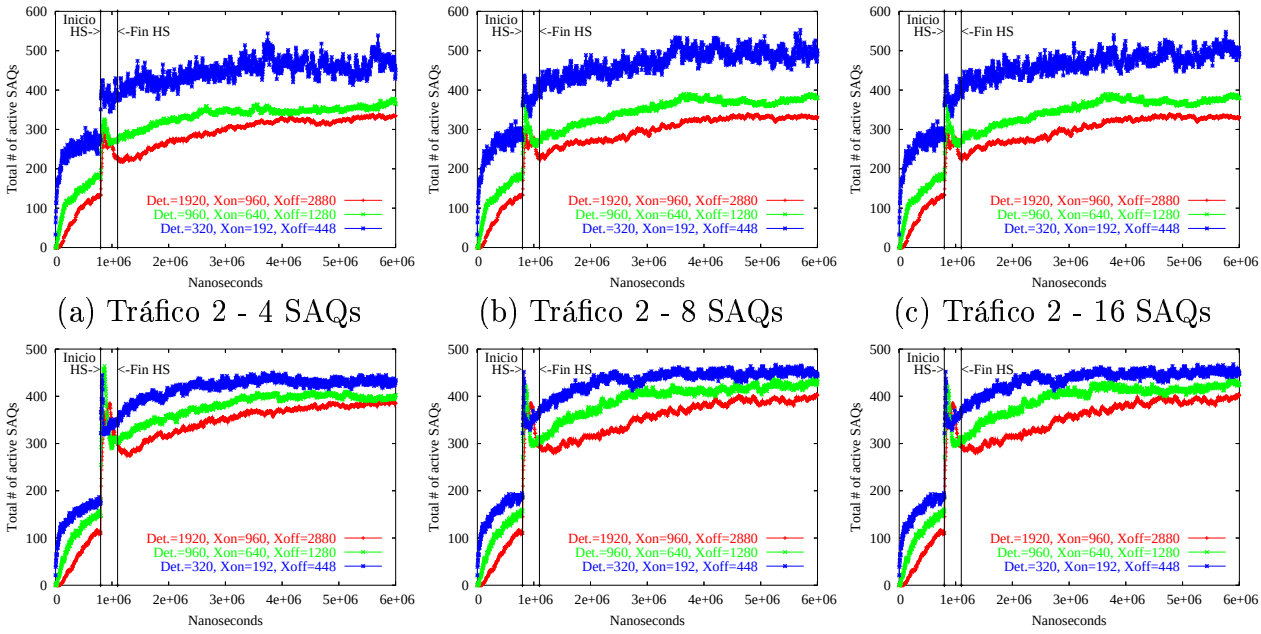
[11] Sin embargo, nótese que, a la vista de los resultados de productividad, el mayor consumo de *SAQs* no parece repercutir en la prestaciones del mecanismo, por lo que el uso de valores de umbrales menores quizá no sea totalmente descartable en algunas circunstancias.



(a) Tráfico 2 - 4 SAQs (b) Tráfico 2 - 8 SAQs (c) Tráfico 2 - 16 SAQs

(d) Tráfico 4 - 4 SAQs (e) Tráfico 4 - 8 SAQs (f) Tráfico 4 - 16 SAQs

Figura 5.20: Número total de *SAQs* activas en la red en función del tiempo para las distintas configuraciones de parámetros *RECN* consideradas, y para los casos de tráfico 2 y 4, *speedup*=1,5.



(a) Tráfico 2 - 4 SAQs (b) Tráfico 2 - 8 SAQs (c) Tráfico 2 - 16 SAQs

(d) Tráfico 4 - 4 SAQs (e) Tráfico 4 - 8 SAQs (f) Tráfico 4 - 16 SAQs

Figura 5.21: Número total de *SAQs* activas en la red en función del tiempo para las distintas configuraciones de parámetros *RECN* consideradas, y para los casos de tráfico 2 y 4, *speedup*=1.

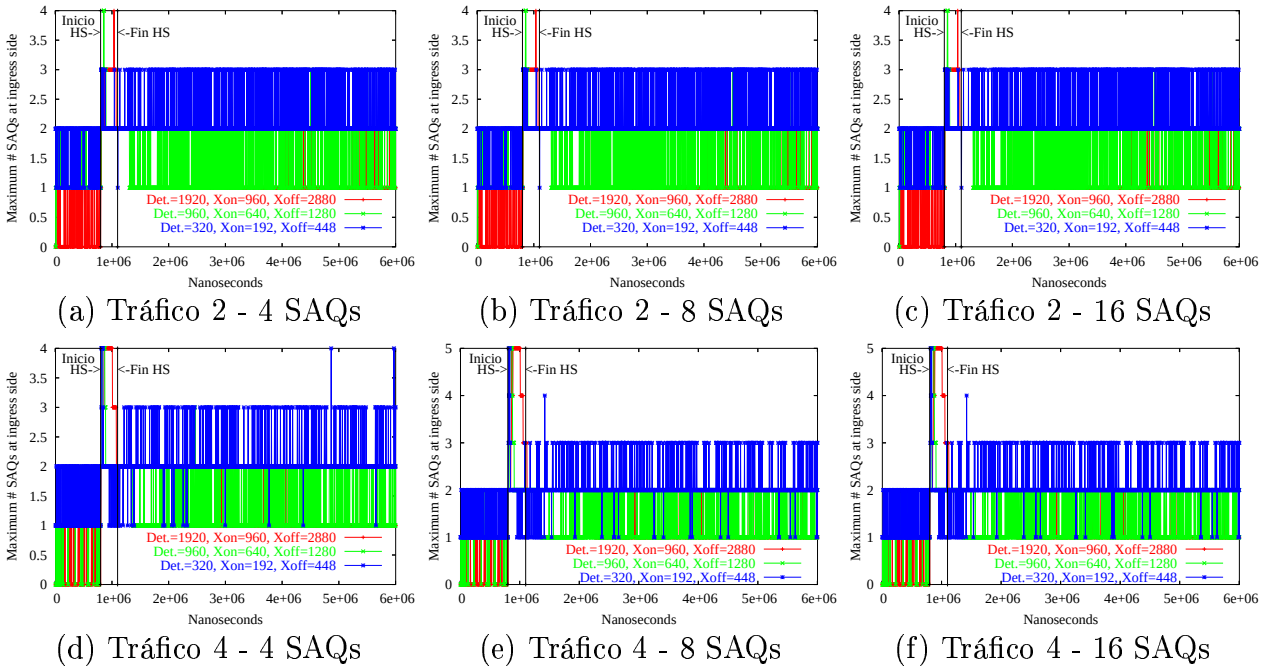
Un dato muy relevante sobre el funcionamiento de la nueva versión de *RECN* respecto a la propuesta inicial se obtiene de la comparación de la figura 5.20 con la figura 4.20. Ambas figuras muestran resultados obtenidos para configuraciones idénticas de red, tráfico, parámetros críticos, y valor de *speedup*, y difieren en la versión de *RECN* utilizada. Puede observarse que el número total de *SAQs* activas en la red es mucho menor (entre la mitad y un tercio) en el caso de la versión mejorada de *RECN*. Por tanto, la nueva versión no sólo resuelve la congestión de forma más eficaz, sino que lo hace utilizando un menor número de *SAQs*. Esto confirma la idea de que los problemas de la propuesta inicial se debían a un uso poco eficiente de las *SAQs* disponibles.

Una vez decidido el uso de la terna de valores medios, sólo queda por decidir el número óptimo de *SAQs* por grupo. Hasta ahora, en ninguna de las figuras anteriores se aprecian diferencias entre los resultados obtenidos para distintos números de *SAQs* por grupo mientras sea para un mismo caso de tráfico, misma terna de valores y mismo valor de *speedup*. Esto, como ya se ha apuntado, llevaría a seleccionar el valor más bajo (4 SAQs) de este parámetro crítico. Para tomar una decisión definitiva en este sentido, recurriremos a los resultados del número máximo de *SAQs* activas en entradas y salidas de los conmutadores.

Las figuras 5.22 y 5.23 muestran el número máximo de *SAQs* activas en una entrada a lo largo del tiempo de simulación, para redes con valor de *speedup*=1,5 en el primer caso y para redes con valor de *speedup*=1 en el segundo. De nuevo, y para no mostrar demasiados resultados, sólo se muestran los obtenidos para los casos de tráfico 2 y 4. Como en casos anteriores, cada gráfica individual muestra los resultados obtenidos para las distintas ternas de valores, y para un único caso de tráfico y un número concreto de *SAQs* por grupo.

A su vez, las figuras 5.24 y 5.25 presentan una estructura idéntica a la descrita en el párrafo anterior, pero mostrando valores del número máximo de *SAQs* activas en una salida (en lugar de en una entrada) en función del tiempo.

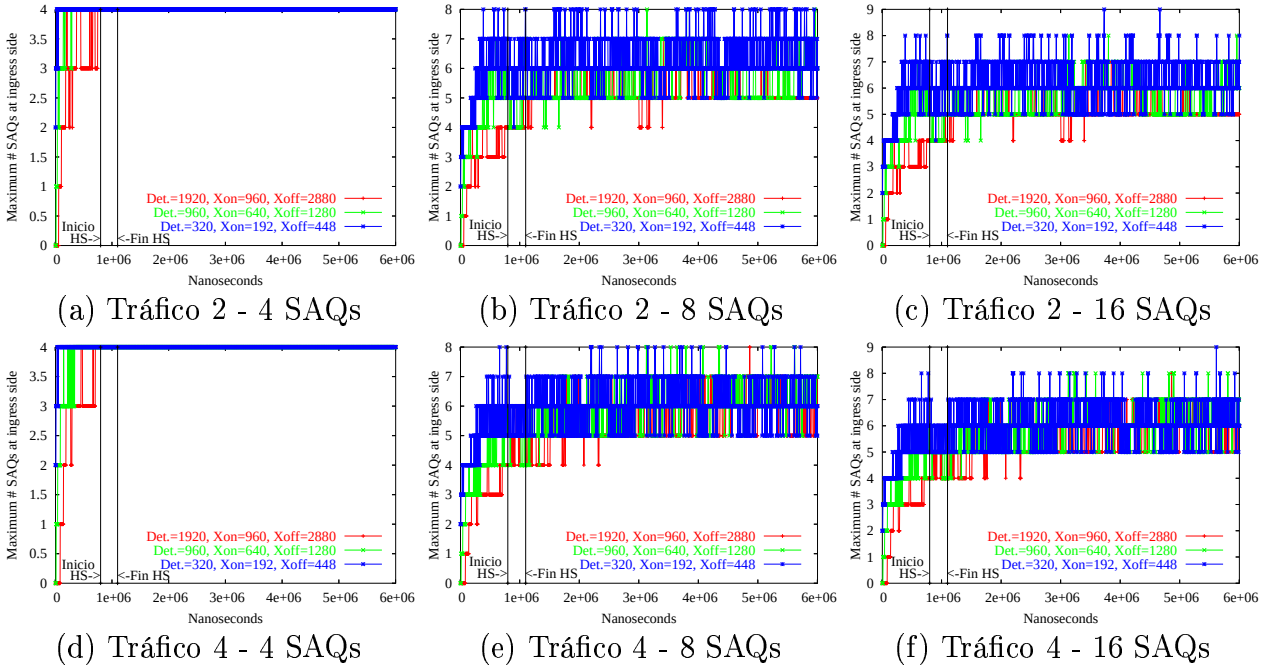
Como primera interpretación de los resultados mostrados en estas figuras, hay que destacar que, efectivamente, en la mayoría de los casos, el mecanismo requiere 4 o menos *SAQs* (tanto en entradas como en salidas) para funcionar correctamente. Sin embargo, para redes con *speedup*=1, las exigencias del mecanismo en cuanto a *SAQs* son superiores, sobre todo en las entradas. Puede comprobarse este hecho en la figura 5.23, donde, para los dos casos de tráfico representados, se consumen más de 4 *SAQs* si está permitido (casos de 8 o 16 *SAQs* por grupo, figuras 5.23.b, 5.23.c, 5.23.e y 5.23.f). En caso de permitir el uso de hasta 4 *SAQs*, los resultados (figuras 5.23.a y 5.23.d) indican un uso constante de este máximo valor. Esto es debido a que, en redes sin aceleración interna (*speedup*=1) en los conmutadores, los paquetes se acumulan rápidamente en las entradas, donde precisamente la nueva versión de *RECN* detecta congestión y asigna *SAQs* automáticamente. Es decir, para un valor de *speedup*=1 las necesidades de *SAQs* del mecanismo serán mayores.



(a) Tráfico 2 - 4 SAQs (b) Tráfico 2 - 8 SAQs (c) Tráfico 2 - 16 SAQs

(d) Tráfico 4 - 4 SAQs (e) Tráfico 4 - 8 SAQs (f) Tráfico 4 - 16 SAQs

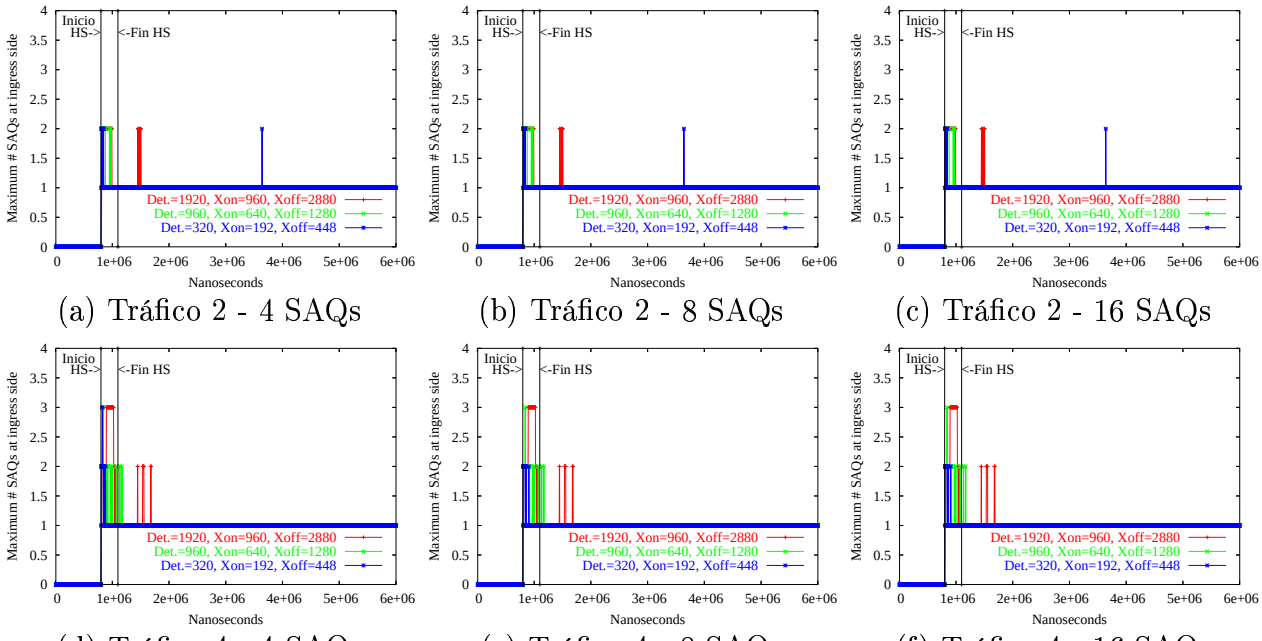
Figura 5.22: Número máximo de *SAQs* activas en una entrada en función del tiempo para las distintas configuraciones de parámetros *RECN* consideradas, y para los casos de tráfico 2 y 4, *speeedup*=1,5.



(a) Tráfico 2 - 4 SAQs (b) Tráfico 2 - 8 SAQs (c) Tráfico 2 - 16 SAQs

(d) Tráfico 4 - 4 SAQs (e) Tráfico 4 - 8 SAQs (f) Tráfico 4 - 16 SAQs

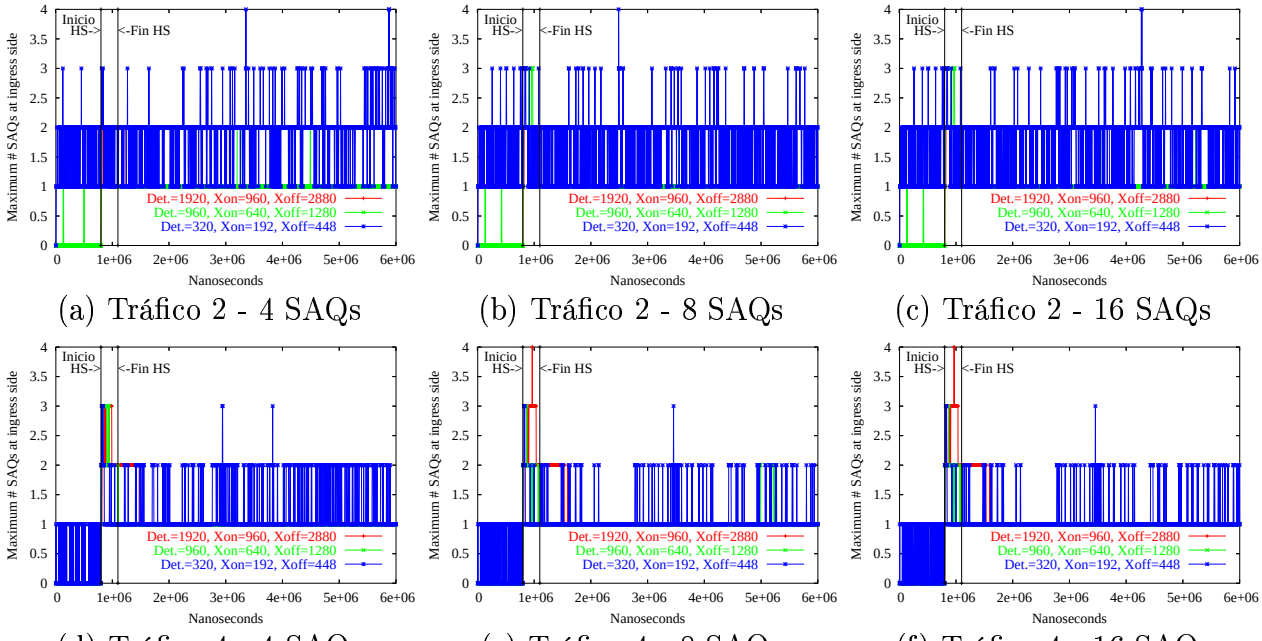
Figura 5.23: Número máximo de *SAQs* activas en una entrada en función del tiempo para las distintas configuraciones de parámetros *RECN* consideradas, y para los casos de tráfico 2 y 4, *speeedup*=1.



(a) Tráfico 2 - 4 SAQs (b) Tráfico 2 - 8 SAQs (c) Tráfico 2 - 16 SAQs

(d) Tráfico 4 - 4 SAQs (e) Tráfico 4 - 8 SAQs (f) Tráfico 4 - 16 SAQs

Figura 5.24: Número máximo de *SAQs* activas en una salida en función del tiempo para las distintas configuraciones de parámetros *RECN* consideradas, y para los casos de tráfico 2 y 4, *speeedup*=1,5.



(a) Tráfico 2 - 4 SAQs (b) Tráfico 2 - 8 SAQs (c) Tráfico 2 - 16 SAQs

(d) Tráfico 4 - 4 SAQs (e) Tráfico 4 - 8 SAQs (f) Tráfico 4 - 16 SAQs

Figura 5.25: Número máximo de *SAQs* activas en una salida en función del tiempo para las distintas configuraciones de parámetros *RECN* consideradas, y para los casos de tráfico 2 y 4, *speeedup*=1.

Sin embargo, los resultados de productividad para los casos de tráfico 2 y 4, valor de *speedup*=1 y 4 *SAQs* como máximo (figuras 5.19.d y 5.19.j) no parecen indicar que la eficacia del mecanismo se resienta por la posible falta de *SAQs* observada. Esto se debe a que esta escasez de *SAQs* se produce realmente sólo en unos cuantos puntos de la red (recordemos que los resultados de máximo número de *SAQs* activas corresponden a una única entrada o salida de toda la red). Por tanto el *HOL blocking* que podría producirse ante la falta de *SAQs* tiene un impacto menor en las prestaciones globales.

Teniendo en cuenta que, en redes mayores, será mayor la cantidad de puntos donde pudiera suceder este efecto de carencia de *SAQs* libres (con el riesgo de *HOL blocking* que esto conlleva), preferimos adoptar una postura "conservadora" y mantener el número máximo de *SAQs* por grupo en 8, que por otra parte no parece un número demasiado elevado. Sin embargo, conviene no olvidar que 4 *SAQs* pueden ser suficientes para afrontar la congestión en numerosas configuraciones, especialmente aquellas redes con valores de *speedup* mayores que 1.

En conclusión, el resultado del ajuste de parámetros críticos para la versión mejorada de *RECN* es similar al obtenido para la versión inicial: es conveniente un máximo de 8 *SAQs* por grupo y el empleo de valores medios para los umbrales de detección, *Xon* y *Xoff*. Sin embargo, nótese que esta decisión se ha tomado a partir de resultados mucho más prometedores en cuanto a eficacia del mecanismo que aquéllos en los que se basaba el ajuste anterior. Obviamente, los valores de los parámetros críticos decididos en el presente ajuste se han empleado para configurar las simulaciones de *RECN* correspondientes a los análisis que se muestran en las siguientes secciones.

5.3.4. Comparación de prestaciones

Una vez establecidos los valores óptimos de los parámetros críticos del mecanismo, podemos comprobar ya si las prestaciones ofrecidas por la nueva versión de *RECN* son suficientes para que el uso de esta técnica presente ventajas respecto al uso de otras técnicas también empleadas para eliminar el *HOL blocking*. Lógicamente, el procedimiento empleado para ello será similar al seguido en el análisis análogo incluido en la evaluación de la propuesta inicial de *RECN* (sección 4.4.4).

Es decir, en el presente análisis, la eficiencia de la versión mejorada de *RECN* se compara, mediante pruebas de distinto tipo, con la eficiencia de las siguientes técnicas: *Virtual Output Queues* a nivel de red (*VOQnet*), *Virtual Output Queues* a nivel de conmutador (*VOQsw*) y canales virtuales. Nótese que esta lista incluye tanto técnicas eficaces pero difícilmente implementables (*VOQnet*) como técnicas implementables pero no plenamente eficaces (*VOQsw* y canales virtuales). Evidentemente, y como ya se ha indicado anteriormente, la bondad de la nueva versión de *RECN* puede estimarse en la medida en que sus prestaciones se acerquen a las obtenidas por (*VOQnet*).

Al igual que en la evaluación comparativa de la propuesta inicial, se ha considerado de ayuda el presentar también en este análisis resultados obtenidos para el caso de no usar ninguna técnica de control de congestión, a modo de referencia para el resto de resultados. Además, también se ha considerado interesante el incluir resultados obtenidos usando encaminamiento adaptativo, ya que, aun sin ser ésta una técnica específicamente destinada a eliminar el *HOL blocking*, también se ha planteado en ocasiones su posible utilidad en este sentido.

5.3.4.1. Configuración de las pruebas

Dada la importancia del presente análisis, se ha procurado que los resultados en los que éste se basa se obtuvieran de un buen número de simulaciones que modelaran entornos muy variados. De hecho, se trata probablemente del análisis más exhaustivo de todos los presentados en esta memoria de tesis. De este modo, las conclusiones que de este análisis pueden extraerse podrán tenerse por ciertas con mayores seguridad y generalidad.

Así, en cuanto al tráfico, se han realizado simulaciones empleando diversos tipos de carga. Por una parte, se han empleado trazas, concretamente las mismas empleadas en análisis anteriores (y cuya procedencia se indica en la sección 4.4.4.1). Cabe recordar que dichas trazas sólo pueden emplearse con redes de hasta 64 terminales, y que, dada su antigüedad, se ha considerado oportuno aplicarles distintos factores de compresión temporal para aumentar la frecuencia de generación de mensajes. Concretamente, se han realizado pruebas con factores de compresión 20 y 40 (los mismos que en la evaluación de la propuesta inicial).

Por otra parte, se han realizado simulaciones empleando distintos patrones de tráfico sintético. En concreto, se han empleado los mismos cuatro casos de tráfico para redes de 64 terminales habitualmente considerados en otros análisis (casos 1 a 4) y que pueden verse en la tabla 5.1. Además, se han empleado nuevos patrones de tráfico sintético, cuyos detalles aparecen en la tabla 5.2. A diferencia de los patrones de tráfico sintético empleados anteriormente, donde la tasa de generación de mensajes es constante para cada caso, en estos nuevos patrones dicha tasa varía incrementalmente, con el objeto de obtener resultados de productividad en función del tráfico generado en lugar de en función del tiempo (tal y como se explicó en la descripción de la herramienta de simulación, sección 4.4.2). Debido a esta diferencia esencial, en los nuevos patrones no existe un intervalo acotado de inyección de tráfico congestionado, sino que éste se inyecta durante toda la simulación (como puede deducirse a partir de la información de la tabla respecto a los instantes de inicio y fin). Como en otros casos de tráfico, en los patrones "incrementales" el tráfico congestionado forma parte de un árbol de congestión formado a partir del envío de paquetes a un destino preferente (*hot-spot*) desde cierto porcentaje de fuentes. Puede comprobarse que los nuevos patrones sí guardan

		Tráfico uniforme			Tráfico árbol de congestión				
Caso de tráfico	Topología	Nº. de fuentes	Destino	Tasa de generación	Nº. de fuentes	Destino	Tasa de generación	Instante inicio	Instante fin
Inc-1	BMIN 64×64 terminales	87,5 %	aleatorio	incremental	12,5 %	Terminal 32	incremental	0 μs	fin sim.
Inc-2	BMIN 64×64 terminales	75 %	aleatorio	incremental	25 %	Terminal 32	incremental	0 μs	fin sim.
M1-Inc-1	malla 256 term. 8×8 conm.	87,5 %	aleatorio	incremental	12,5 %	Terminal 32	incremental	0 μs	fin sim.
M1-Inc-2	malla 256 term. 8×8 conm.	75 %	aleatorio	incremental	25 %	Terminal 32	incremental	0 μs	fin sim.
M2-Inc-1	malla 256 term. 4×4 conm.	87,5 %	aleatorio	incremental	12,5 %	Terminal 32	incremental	0 μs	fin sim.
M2-Inc-2	malla 256 term. 4×4 conm.	75 %	aleatorio	incremental	25 %	Terminal 32	incremental	0 μs	fin sim.

Tabla 5.2: Tráficos "incrementales" empleados en la comparación de prestaciones.

similitudes con los anteriores en cuanto a las proporciones entre el número de fuentes que inyecta tráfico uniforme o congestionado.

Puede observarse ya en la tabla 5.2 que algunos de los nuevos casos de tráfico sintético "incremental" se han aplicado a la habitual topología multietapa bidireccional (*BMIN*) de 64 terminales, pero otros lo han sido a distintas configuraciones de redes con topología de malla, lo que supone una novedad en cuanto a la estructura de la red. Las dos configuraciones de malla empleadas constan de 256 terminales, conectados en el primer caso mediante 64 conmutadores (malla 8×8, con 4 terminales por conmutador) y mediante 16 conmutadores en el segundo (malla 4×4, con 16 terminales por conmutador). Obviamente, en estos casos tanto el número como la organización de los enlaces de los conmutadores son diferentes a los correspondientes a la *BMIN*: cada conmutador se conecta a otros 4 conmutadores (a excepción de los conmutadores situados en los bordes de la malla), y todos los conmutadores tienen terminales conectados (4 ó 16 dependiendo de la configuración de la red). Es decir, en el caso de las mallas el número de puertos de cada conmutador puede variar entre 6 y 20.

En lo que respecta a los enlaces se sigue asumiendo un único ancho de banda de 8 *Gbps*. Tampoco se ha variado el tamaño de las memorias en cada entrada o salida (32 KB). En cambio, sí se han realizado simulaciones usando diferentes valores de *speedup* (1,5 y 1), habida cuenta de los diferentes resultados obtenidos hasta ahora para estos valores.

En cuanto a los *Input Adapters*, se siguen asumiendo las mismas características consideradas en todos los análisis anteriores: 32 KB de memoria en sus salidas y división de los mensajes generados en paquetes de 64 bytes[12]. Cabe recordar que la memoria de los IAs debe tener la misma organización establecida en las memorias de los puertos

[12]Merece la pena comentar que se han realizado simulaciones con tamaños de paquete mayores (512 bytes), pero, al ser los resultados cualitativamente idénticos a los obtenidos con 64 bytes, no se ha considerado oportuno incluirlos en este documento.

de salida de los conmutadores, y dicha organización, a su vez, está en función de la técnica de eliminación del *HOL blocking* simulada.

En este sentido, y teniendo en cuenta el planteamiento del presente análisis, era obligatorio que se realizaran simulaciones empleando distintas técnicas de eliminación del *HOL blocking*. En las pruebas en las que se ha empleado *RECN*, como ya se ha comentado, el mecanismo se ha configurado con los valores de parámetros críticos decididos tras el ajuste realizado en el análisis previo (valores medios de los umbrales y 8 *SAQs* por grupo). En este caso, debe tenerse en cuenta que, en las entradas de los conmutadores, *RECN* necesita que existan tantas colas de detección como salidas tenga el conmutador. Por tanto, se han establecido 8 colas de detección en cada entrada en el caso de las redes $BMIN$, y un número variable de colas de detección (entre 6 y 20) en cada entrada en el caso de las mallas, dependiendo de las dimensiones de la malla y de la posición del conmutador en la misma.

Cuando se ha simulado *VOQnet* como técnica de eliminación del *HOL blocking*, la memoria de datos en entradas y salidas se ha dividido en tantas colas como destinos existan en la red, lo que supone 64 colas para las redes $BMIN$ y 256 colas para las dos configuraciones de malla. Para *VOQsw*, el número de colas en entradas y salidas debe ser el mismo que el número de puertos del conmutador, por lo que se han establecido 8 colas para entradas y salidas de conmutadores de redes $BMIN$, y entre 6 y 20 colas para las entradas y salidas de los conmutadores de las mallas (por los motivos ya expuestos). En las simulaciones de canales virtuales se han asumido 4 colas por puerto (4 *Virtual Channels*) con política de almacenamiento de paquetes en la cola más vacía. También, como ya se ha apuntado, se han realizado pruebas sin utilizar ninguna técnica de control de congestión, en cuyo caso existe una única cola en entradas y salidas. Igualmente, en las pruebas realizadas usando encaminamiento adaptativo, las entradas y salidas disponen de una sola cola.

Respecto al criterio concreto empleado para implementar dicho encaminamiento adaptativo, se ha escogido el de encaminar cada paquete en cabeza de una cola de entrada hacia la salida del conmutador con la cola más vacía. Teóricamente, esto contribuiría a aliviar la congestión en las colas y, en cierta medida, a eliminar el *HOL blocking*. Cabe recordar que en redes multietapa la adaptatividad en cuanto al encaminamiento sólo es posible en las etapas iniciales de una ruta, por lo que sólo en estas etapas se ha aplicado el mencionado criterio (en las etapas finales el encaminamiento debe ser determinista para alcanzar el destino correspondiente).

En lo referente a las métricas, se ha considerado de nuevo la productividad de la red como principal medida de la eficacia de las distintas técnicas para eliminar el *HOL blocking*. Ahora bien, en esta ocasión los resultados de productividad se han obtenido tanto en función del tiempo como en función de la carga de tráfico (dependiendo del caso de tráfico empleado en la simulación). Además, como métrica adicional, se han obtenido resultados de latencia de los mensajes (latencia de red), de cara a analizar las

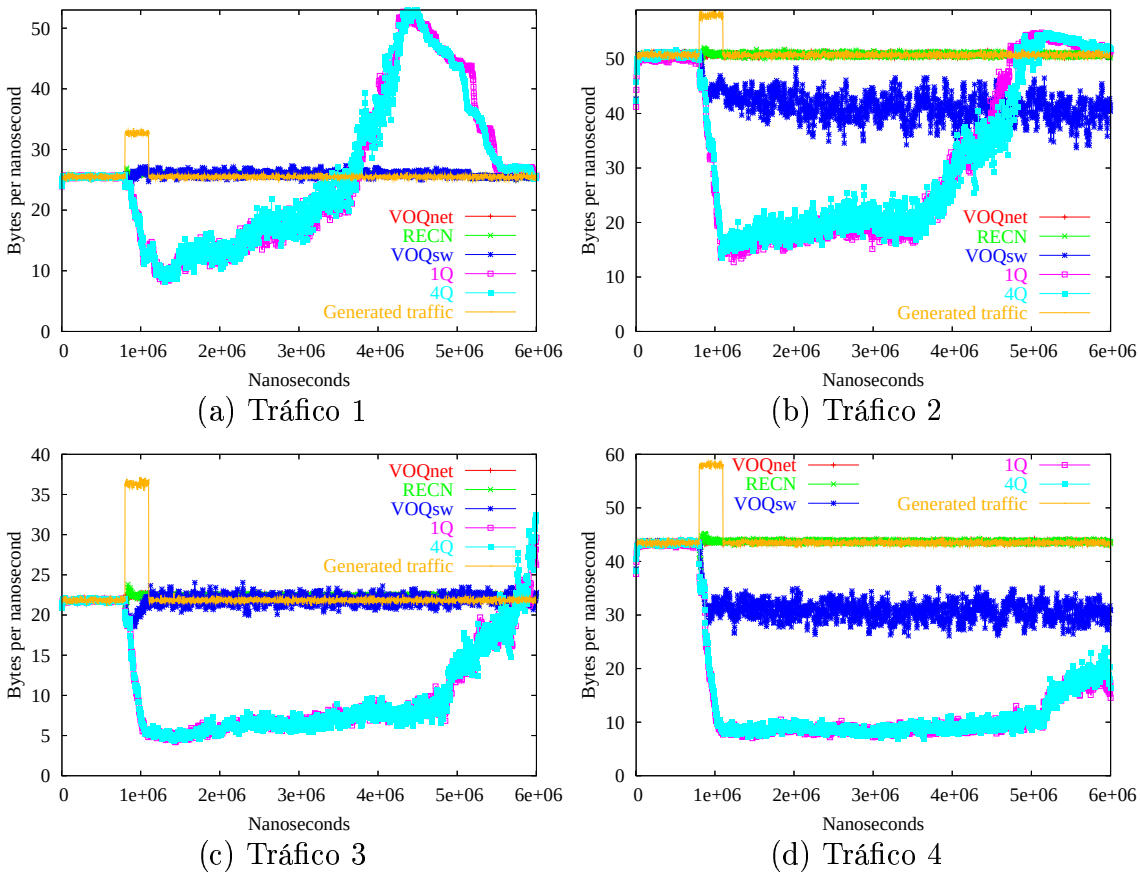


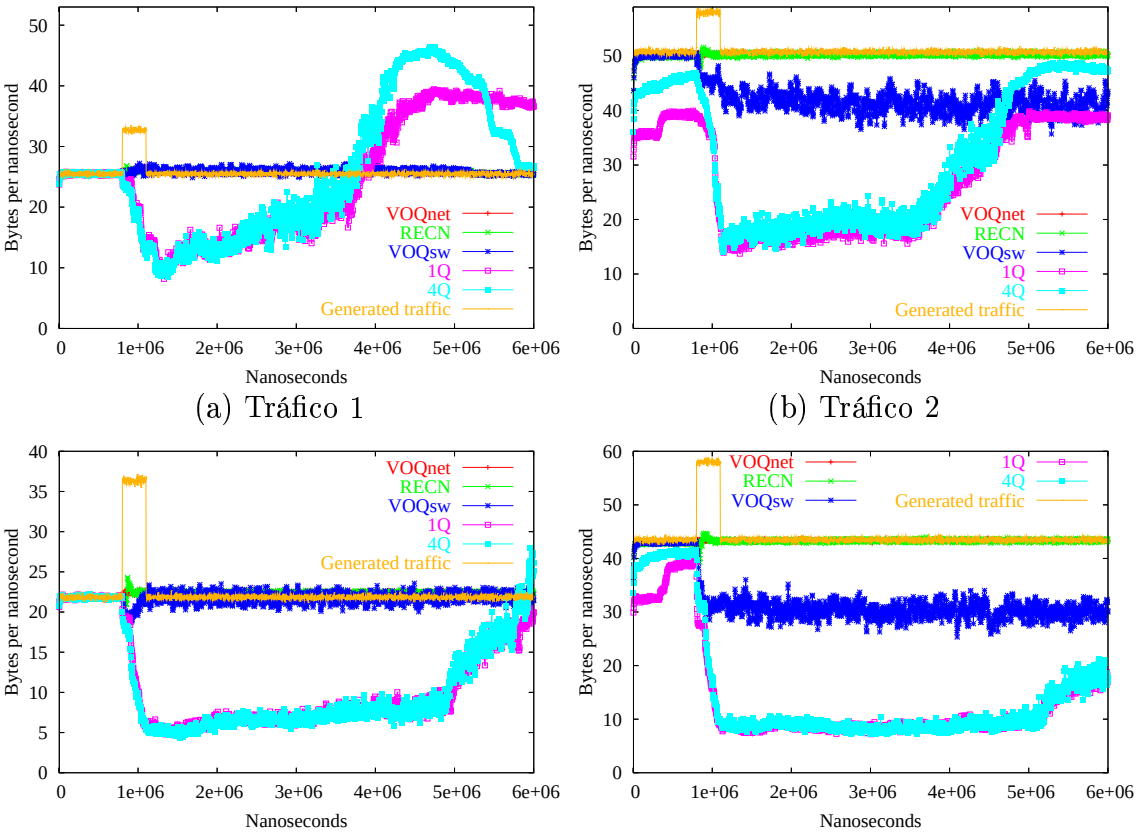
Figura 5.26: Productividad de la red en función del tiempo para los casos de tráfico sintético 1 a 4 y para las distintas técnicas de control de congestión consideradas. Conmutadores con *speedup*=1,5.

prestaciones de las distintas técnicas desde otro punto de vista. La latencia también se ha medido bien en función del tiempo, bien en función de la carga de tráfico. En ambos casos, sólo se ha considerado la latencia de los paquetes pertenecientes a flujos no congestionados, ya que de otro modo esta medida se desvirtúa por los grandes retardos que sufren inevitablemente los paquetes congestionados.

En resumen, en la siguiente sección se ofrecen resultados de productividad de la red latencia de los mensajes, en ambos casos en función del tiempo y de la carga de tráfico, para distintas combinaciones de los casos de tráfico, valores de *speedup*, y técnicas de eliminación del *HOL* blocking indicados en la presente sección.

5.3.4.2. Resultados

Las figuras 5.26 y 5.27 muestran resultados de productividad en función del tiempo para los casos de tráfico sintético 1 a 4. La primera de estas figuras muestra resultados



(a) Tráfico 1 (b) Tráfico 2

(c) Tráfico 3 (d) Tráfico 4

Figura 5.27: Productividad de la red en función del tiempo para los casos de tráfico sintético 1 a 4 y para las distintas técnicas de control de congestión consideradas. Conmutadores con *speedup*=1.

para un valor de *speedup* de los conmutadores de 1,5, mientras que para los mostrados en la segunda, este valor es de 1 (conmutadores sin aceleración interna). En ambas figuras se muestran, en cada gráfica individual, varias series de resultados que corresponden a las distintas técnicas de eliminación del *HOL blocking* consideradas: *RECN*, *VOQnet*, *VOQsw*, canales virtuales (4Q) y ninguna técnica (1Q). También, como es habitual, en cada gráfica aparece una sexta serie (en amarillo) representando el tráfico inyectado en función del tiempo.

Por el análisis anterior, ya conocíamos que *RECN* es capaz de eliminar de forma prácticamente total el *HOL blocking* para estos casos de tráfico y para ambos valores de *speedup*, ya que, como también puede apreciarse en estas nuevas figuras, la productividad obtenida usando *RECN* se ajusta muy estrechamente a la máxima posible en todos los casos. Este máximo también es alcanzado por *VOQnet*, como no podía ser de otro modo, para todas las configuraciones (de hecho, los resultados de *RECN*

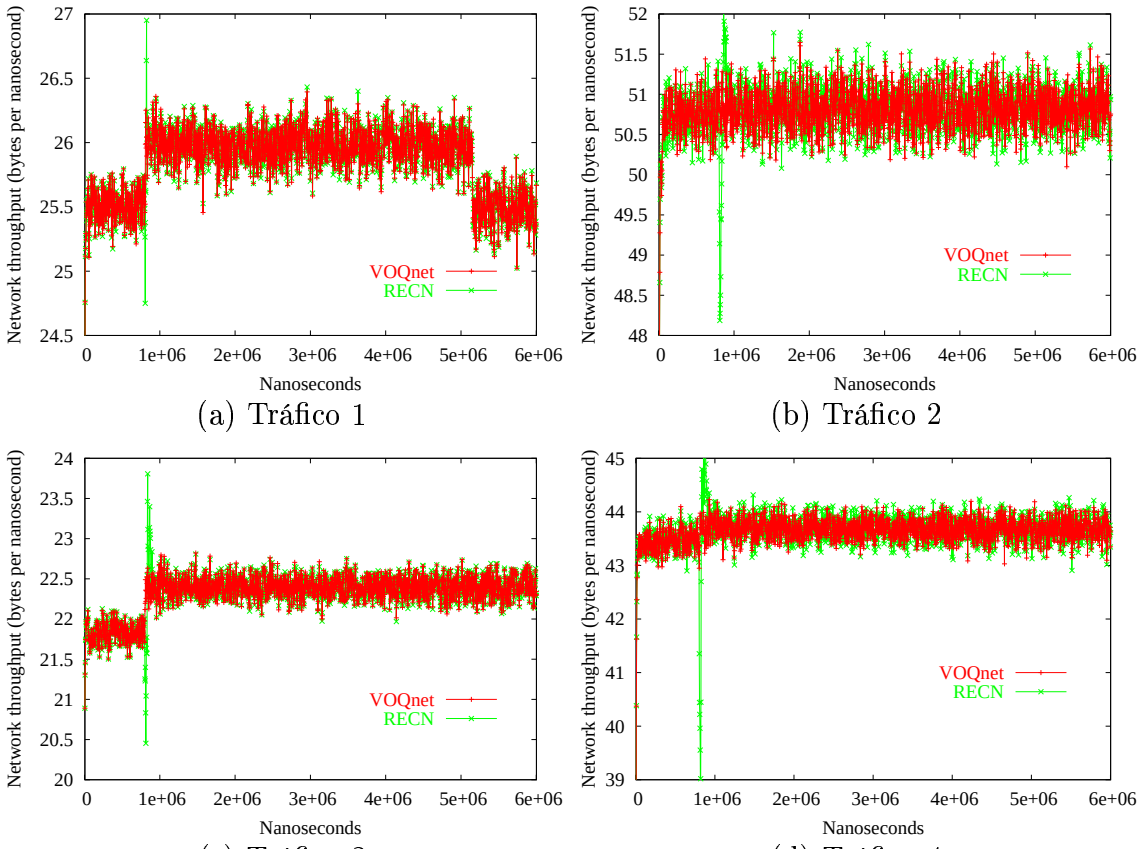


Figura 5.28: Productividad de la red (zoom) en función del tiempo para *VOQnet* y *RECN* y para los casos de tráfico sintético 1 a 4. Conmutadores con *speedup*=1,5.

y *VOQnet* son tan parecidos que, en estas figuras, las series de resultados correspondientes a *VOQnet* y *RECN* se superponen). Para otras técnicas, los resultados son mucho peores en los casos de tráfico más intenso (casos 2 y 4), independientemente del valor de *speedup*. Concretamente, la productividad obtenida usando *VOQsw* llega a caer hasta un 30 % respecto al nivel alcanzado por *RECN* para el caso de tráfico 4 y ambos valores de *speedup*. Por otra parte, la productividad obtenida usando canales virtuales llega a caer hasta un 80 % respecto al nivel máximo. Como es conocido, estos malos resultados se deben a que estas últimas técnicas no eliminan completamente el *HOL blocking*.

Como se ha mencionado, los buenos resultados obtenidos por *RECN* llevan a que en las gráficas anteriores, estas series de resultados se solapen con las correspondientes a *VOQnet*. Para apreciar mejor hasta qué punto los resultados son similares para ambas técnicas, las figuras 5.28 y 5.29 muestran los mismos resultados de *VOQnet* y *RECN* mostrados anteriormente, pero en esta ocasión sin mostrar resultados de ninguna otra técnica de control de congestión. Además, y para apreciar mejor las posibles diferencias,

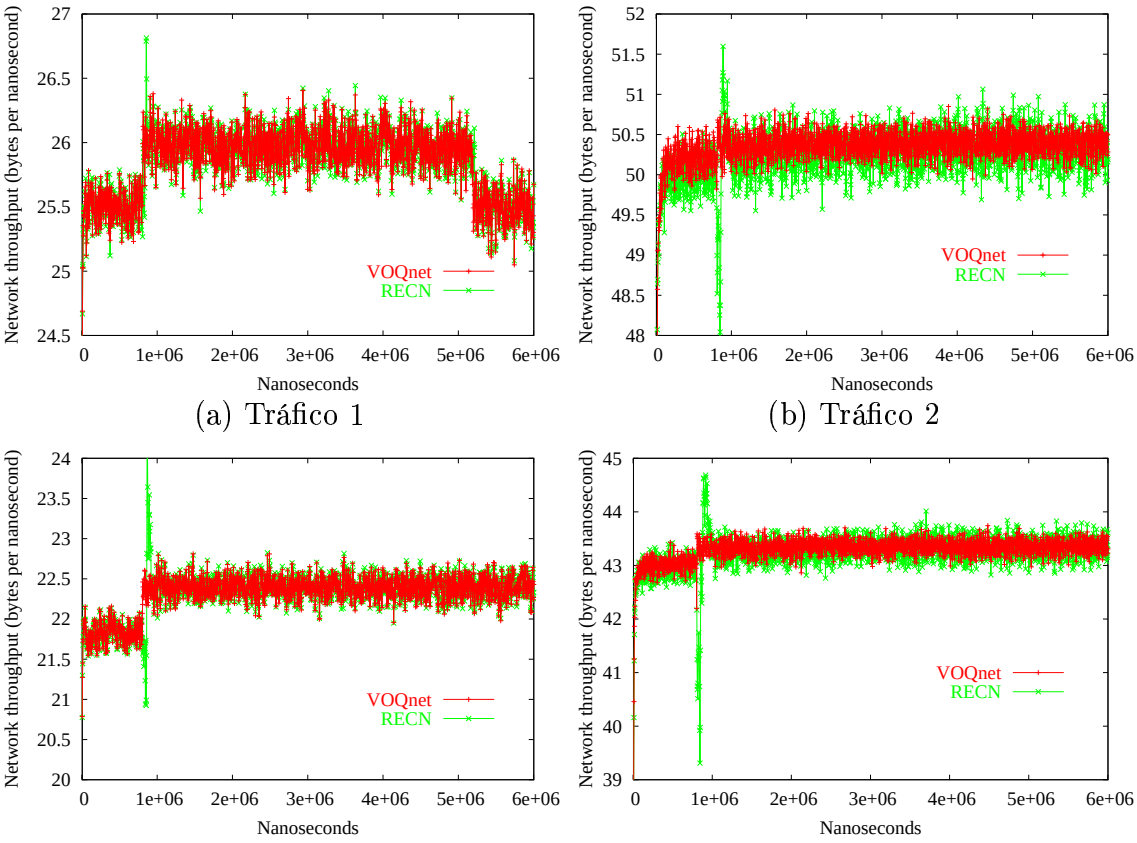
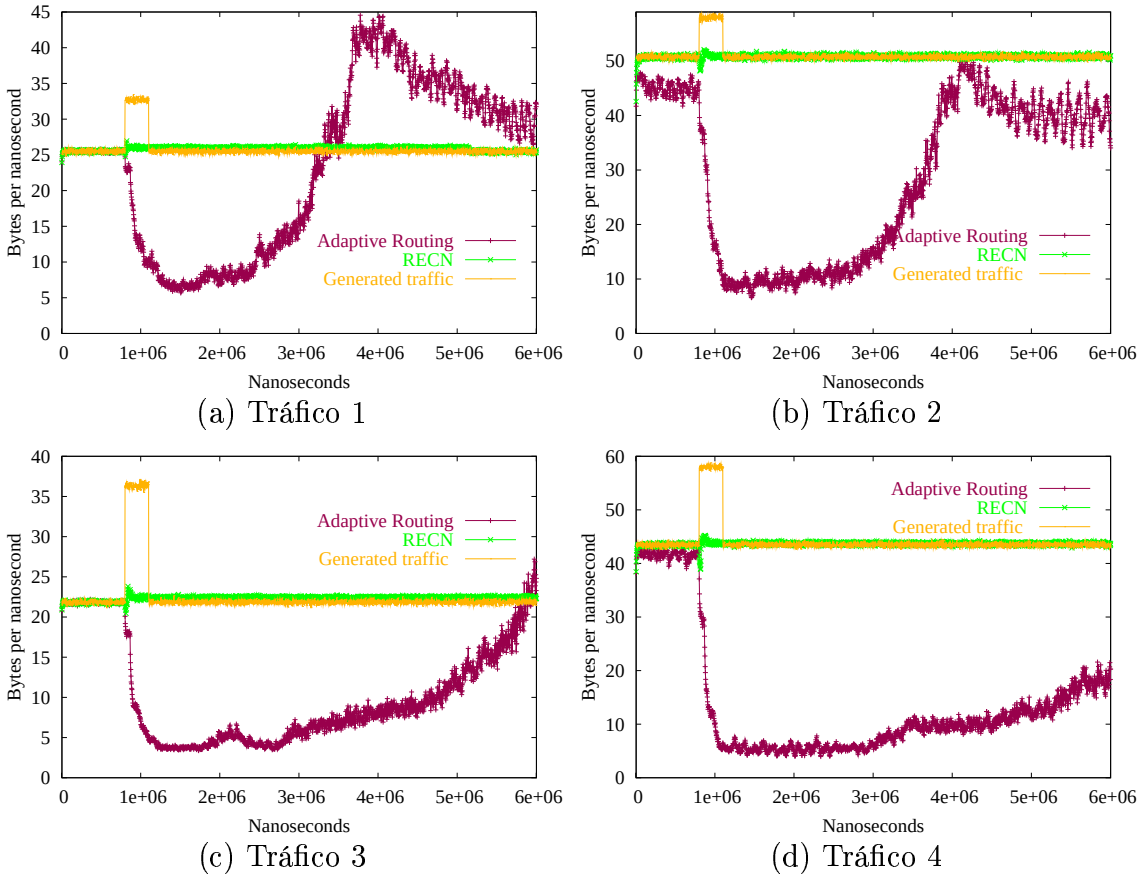


Figura 5.29: Productividad de la red (zoom) en función del tiempo para *VOQnet* y *RECN* y para los casos de tráfico sintético 1 a 4. Conmutadores con *speedup*=1.

se ha reducido el intervalo de valores de productividad representados en las figuras (es decir, se trata de un "zoom" sobre el eje de ordenadas).

En estas últimas figuras puede apreciarse más claramente que los resultados de *RECN* son efectivamente casi idénticos a los de *VOQnet*. La única diferencia digna de mención es el momentáneo descenso de la productividad que se produce para *RECN*, justo cuando comienza a inyectarse el tráfico congestionado. Este descenso es debido al tiempo que tarda el mecanismo en detectar congestión y activarse (o sea, el tiempo que se tarda en asignar *SAQs* para el árbol de congestión). De todos modos, puede observarse que la duración de esta bajada de productividad es mínima, y en ningún caso la caída es superior al 10 % del máximo alcanzable.

Siguiendo con resultados de productividad en función del tiempo, la figura 5.30 compara los obtenidos usando *RECN* con los obtenidos usando encaminamiento adaptativo, para los casos de tráfico sintético 1 a 4. En todos los casos el valor de *speedup* de los conmutadores es de 1,5. Como puede comprobarse, el uso de encaminamiento adaptativo (recordemos que basado en seleccionar la salida con la cola más vacía)



(a) Tráfico 1 (b) Tráfico 2

(c) Tráfico 3 (d) Tráfico 4

Figura 5.30: Productividad de la red en función del tiempo para encaminamiento adaptativo y *RECN* y para los casos de tráfico sintético 1 a 4. Conmutadores con *speedup*=1,5.

no consigue en ningún caso solucionar la congestión presente en la red, descendiendo drásticamente la productividad de la misma (en torno a un 80 % respecto al máximo alcanzable) en cuanto comienza a inyectarse tráfico congestionado. Obviamente, las diferencias con los resultados obtenidos con *RECN* son enormes, lo que nos permite afirmar que nuestra técnica elimina el *HOL blocking* mucho más eficazmente que el encaminamiento adaptativo.

Es evidente que, de todas estas comparaciones de resultados de productividad de la red, *RECN* sale excepcionalmente bien parado. Por una parte consigue, en todos los casos, las mismas prestaciones obtenidas con *VOQnet*, que en la práctica es una técnica casi imposible de implementar debido a sus altos requisitos en cuanto a recursos. Nótese que, con la configuración de red empleada para obtener estos resultados, *VOQnet* necesitaría 64 colas en entradas y salidas, mientras que *RECN* necesita sólo 16 colas en las entradas y 7 en las salidas. Por otra parte, *RECN* consigue eliminar el *HOL blocking* mucho más eficazmente que *VOQsw* o canales virtuales, que sí son técnicas más

o menos implementables. En definitiva, y al menos para los casos mostrados, *RECN* consigue las prestaciones de una técnica "ideal" utilizando muchos menos recursos que ésta, y supera de largo a otras técnicas poco exigentes en cuanto a recursos.

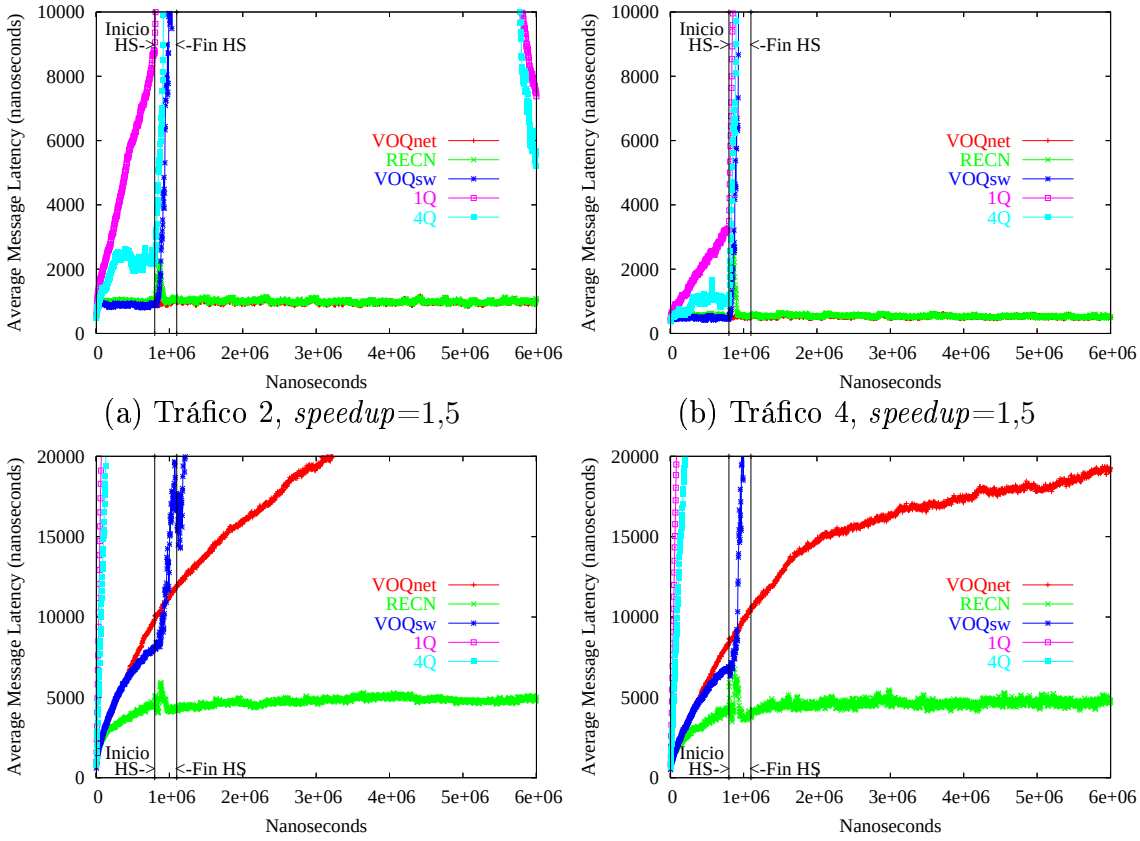
Nótese que la anterior conclusión es válida independientemente del valor de *speedup*, como puede comprobarse en las figuras. Esto supone una mejora fundamental en *RECN*, ya que, con la propuesta inicial del mecanismo, no se conseguían buenos resultados cuando se aplicaba en redes formadas por conmutadores con valor de *speedup*=1. En este sentido, es particularmente interesante comparar la figura 5.27 con la figura 4.24, que muestra resultados para los mismos casos de tráfico y valor de *speedup* (1), pero considerando la propuesta inicial de *RECN*. En esta figura del capítulo anterior, *RECN* apenas superaba a *VOQsw* (en algunos casos, sucedía incluso lo contrario), mientras que ahora los resultados obtenidos por *RECN* son mucho mejores que los de *VOQsw* en todos los casos. Esto indica que los cambios introducidos en *RECN* eran sin duda necesarios para el buen funcionamiento del mecanismo, sobre todo en redes con conmutadores sin aceleración interna.

La siguiente figura en este análisis (figura 5.31) muestra resultados de latencia de los mensajes en función del tiempo, para los casos de tráfico sintético más intenso (casos 2 y 4) y para los dos valores de *speedup* considerados. Como es habitual, cada gráfica individual contiene varias series de resultados, correspondientes a las distintas técnicas de eliminación del *HOL blocking* consideradas. Además, cada gráfica incluye dos marcas que indican el principio y el fin de la inyección de tráfico congestionado.

Estos resultados de latencia ratifican las buenas prestaciones de *RECN* observadas para la productividad. En todos los casos, *RECN* consigue los mejores resultados de latencia de todas las técnicas empleadas. Puede observarse que *RECN* es capaz de mantener la latencia en un nivel prácticamente constante a lo largo de todo el intervalo de simulación, independientemente del caso de tráfico y del valor de *speedup*. Los resultados son escandalosamente peores para *VOQsw* y canales virtuales, donde la latencia se dispara hasta niveles altísimos[13]. Más aún, en redes con *speedup*=1, *RECN* consigue mejores resultados que *VOQnet* (figuras 5.31.c y 5.31.d).

Esta ventaja de *RECN* sobre *VOQnet* es debida a que esta última técnica separa todos los flujos de datos, pero no detecta realmente cuáles son flujos congestionados, mientras que *RECN* sí identifica los flujos congestionados, los almacena en *SAQs* y da a éstas menor preferencia en el acceso al *crossbar* y a los enlaces de salida. Es decir: con *RECN*, el árbitro del conmutador y los de los enlaces pueden "marginar" a los flujos congestionados al estar éstos identificados, mientras que esto no es posible en *VOQnet*. En consecuencia, *VOQnet* no puede evitar que la latencia se degrade ligeramente cuando los paquetes congestionados ocupen el *crossbar* e impidan cruzar a los paquetes fríos situados en colas del mismo puerto. Este efecto es especialmente

[13]El nivel máximo alcanzado en estos casos ni siquiera se muestra en las gráficas; de otro modo no podrían apreciarse los valores de latencia obtenidos para *RECN*.



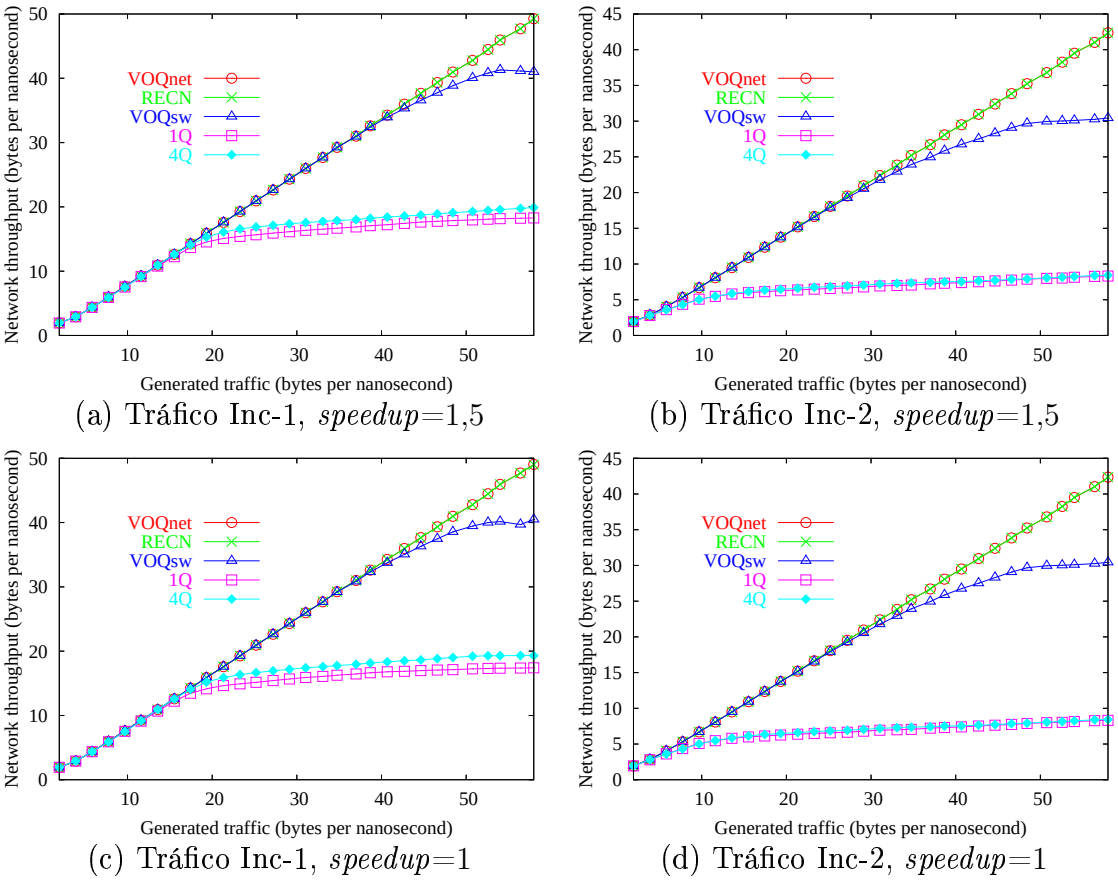
(a) Tráfico 2, *speedup*=1,5 (b) Tráfico 4, *speedup*=1,5

(c) Tráfico 2, *speedup*=1 (d) Tráfico 4, *speedup*=1

Figura 5.31: Latencia media de los mensajes no congestionados en función del tiempo para los casos de tráfico sintético 2 (a y c) y 4 (b y d) y para las distintas técnicas de control de congestión consideradas. Conmutadores con *speedup*=1,5 (a y b) y *speedup*=1 (c y d).

apreciable en redes sin aceleración interna de los conmutadores, donde los paquetes son extraídos de las entradas más lentamente, y por tanto, los flujos congestionados ocuparán durante algo más de tiempo el *crossbar* cuando se les conceda cruzar. Ahora bien, las consecuencias de este efecto no son desastrosas debido a que la mayoría del tiempo los paquetes congestionados estarán "frenados" por el propio control de flujo, sin posibilidad de solicitar el *crossbar*.

Siguiendo con las comparaciones, la figura 5.32 muestra de nuevo la productividad de la red, en esta ocasión como una función del tráfico generado. Concretamente se muestran resultados para los casos de tráfico sintético Inc-1 e Inc-2 (recordemos que ambos casos asumen que la red es una $BMIN$ de 64 terminales), y para valores de *speedup* de 1,5 y 1. Como siempre, cada gráfica individual incluye una serie de resultados para cada una de las distintas técnicas de eliminación del *HOL blocking*.



(a) Tráfico Inc-1, *speedup*=1,5

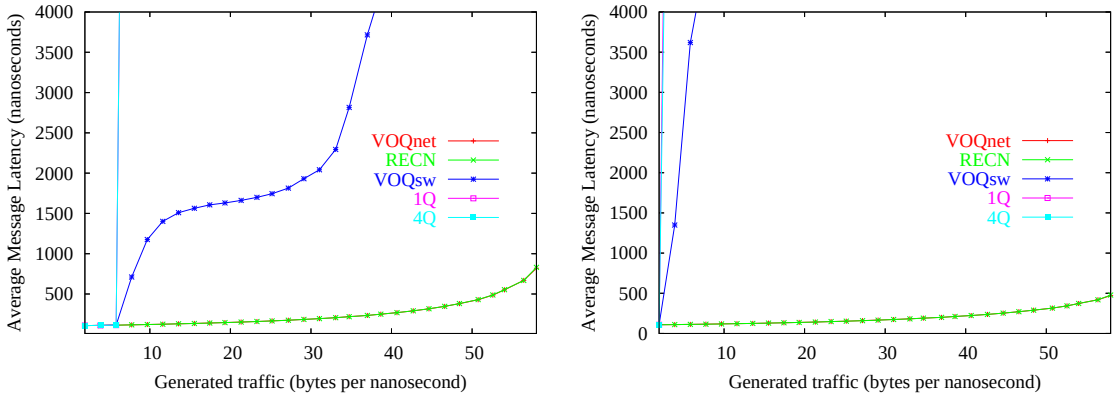
(b) Tráfico Inc-2, *speedup*=1,5

(c) Tráfico Inc-1, *speedup*=1

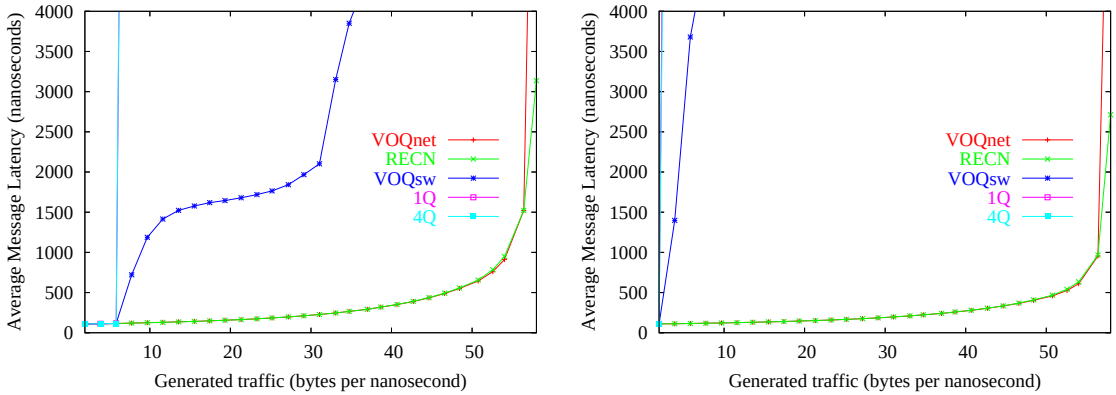
(d) Tráfico Inc-2, *speedup*=1

Figura 5.32: Productividad de la red en función del tráfico generado para los casos de tráfico sintético Inc-1 e Inc-2, y para conmutadores con *speedup*=1,5 y *speedup*=1.

Aunque la información recogida en la figura 5.32 es distinta a la contenida en las anteriores, las conclusiones que de ella pueden extraerse son similares. Así, en todos los casos, *RECN* obtiene unas prestaciones que igualan las de *VOQnet* y superan a las del resto de técnicas, y mantiene la productividad al máximo, independientemente de la carga, gracias a que elimina totalmente el *HOL blocking*. Precisamente, estas gráficas resultan especialmente interesantes para corroborar que, sin una correcta eliminación del *HOL blocking*, la red entra en saturación (la productividad no aumenta a pesar de generarse más tráfico) para cargas de tráfico mucho menores. Este efecto es drástico para el caso de canales virtuales, que no pueden evitar que la red entre en saturación para cargas de tráfico muy bajas. Por su parte, *VOQsw* entra en saturación para cargas medias de tráfico. Esto refuerza nuestra idea sobre la necesidad de técnicas de eliminación del *HOL blocking* realmente eficientes, que permitan sacar el máximo partido de la red, incluso con cargas muy altas de tráfico.



(a) Tráfico Inc-1, *speedup*=1,5 (b) Tráfico Inc-2, *speedup*=1,5



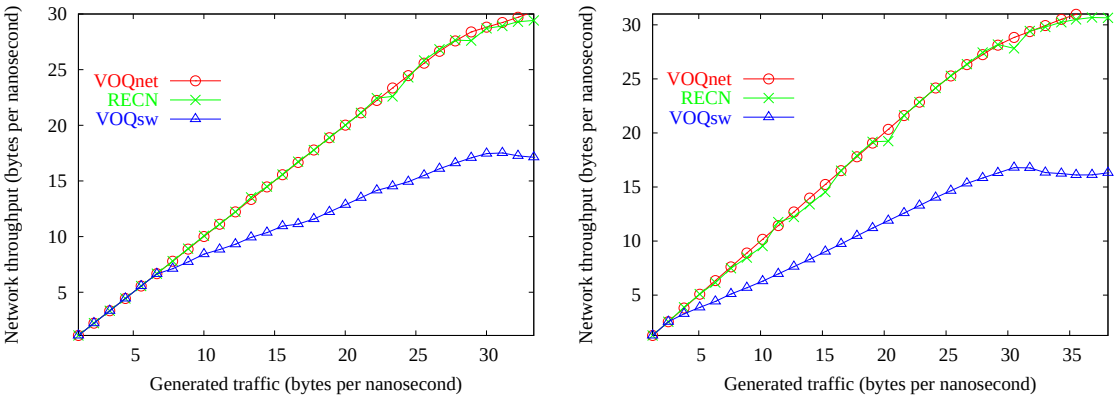
(c) Tráfico Inc-1, *speedup*=1 (d) Tráfico Inc-2, *speedup*=1

Figura 5.33: Latencia media de los mensajes no congestionados en función del tráfico generado para los casos de tráfico sintético Inc-1 y Inc-2, y para conmutadores con *speedup*=1,5 y *speedup*=1.

La figura 5.33 complementa a la anterior figura, mostrando, para los mismos casos de tráfico (Inc-1 e Inc-2) y valores de *speedup* (1,5 y 1), resultados de latencia de los mensajes en función del tráfico generado.

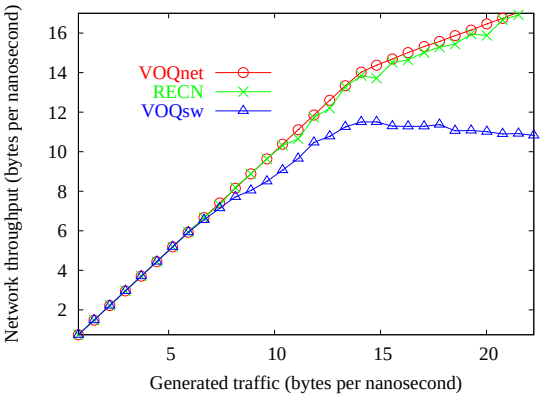
Como puede apreciarse en esta figura, también la latencia indica que la red entrará en saturación para cargas bajas de tráfico si el *HOL blocking* no es eliminado correctamente. Así lo demuestra el brusco crecimiento de latencia que puede observarse para *VOQsw* y canales virtuales cuando la carga es todavía bastante modesta[14]. Las técnicas que sí eliminan completamente el *HOL blocking* (*VOQnet* y *RECN*), en cambio, mantienen la latencia a niveles mucho más bajos, apreciándose un incremento muy lento de la misma al ir aumentando la carga de tráfico. Sólo para cargas muy altas de tráfico se observa un incremento notable de la latencia para *VOQnet* y *RECN*, y sólo para el caso de tráfico Inc-1 y con valor de *speedup*=1 en los conmutadores (figura

[14]De nuevo, los niveles máximos alcanzados en estos casos no se muestran, en beneficio de una mejor visibilidad del resto de resultados.

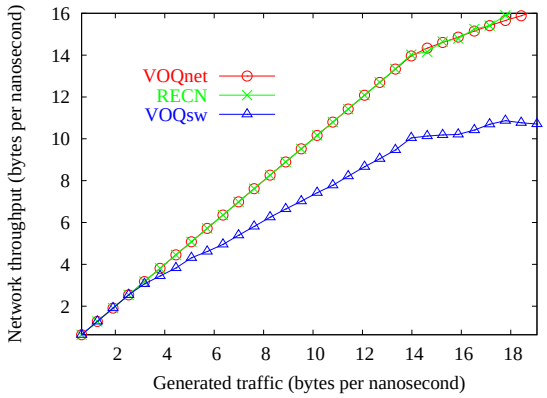


(a) Tráfico M1-Inc-1 (malla 8×8)

(b) Tráfico M1-Inc-2 (malla 8×8)



(c) Tráfico M2-Inc-1, (malla 4×4)



(d) Tráfico M2-Inc-2, (malla 4×4)

Figura 5.34: Productividad de la red (mallas) en función del tráfico generado, para los casos de tráfico sintético M1-Inc-1, M1-Inc-2, M2-Inc-1 y M2-Inc-2.

5.33.c). Nótese que de nuevo en esta figura se observa una ligera ventaja de *RECN* sobre *VOQnet*, debido a las ya comentadas diferencias de ambas técnicas en cuanto al arbitraje sobre los paquetes congestionados.

Continuando con los resultados para tráficos "incrementales", la figura 5.34 muestra resultados de productividad de la red en función del tráfico generado para los casos de tráfico sintético M1-Inc-1, M1-Inc-2, M2-Inc-1 y M2-Inc-2. Como ya se ha mencionado, en todos estos casos se asume que la red tiene topología de malla, con 8×8 conmutadores para los casos M1 y con 4×4 conmutadores para los casos M2, pero siempre con 256 terminales. En todos las simulaciones el *speedup* de los conmutadores se ha fijado en 1,5. Cada gráfica individual contiene tres series de resultados que corresponden respectivamente a *VOQnet*, *RECN* y *VOQsw*.

La primera conclusión que puede extraerse de los resultados mostrados en esta figura es que *RECN* se comporta en topologías de malla al mismo nivel que en topologías de tipo *BMIN*. En todos los casos de tráfico mostrados, la productividad alcanzada usando *RECN* es prácticamente idéntica a la conseguida usando *VOQnet*, y claramente

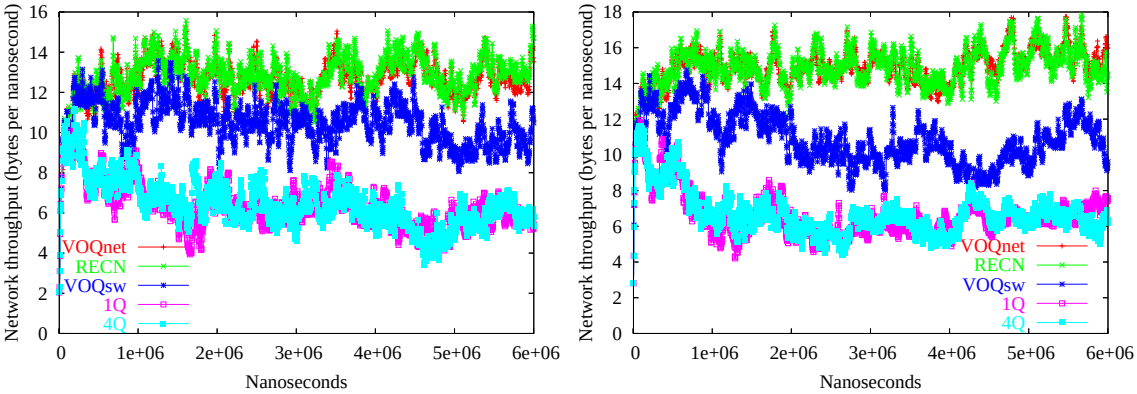
superior a los niveles máximos a los que se llega con *VOQsw*. Es decir, no parece que la topología de la red suponga un impedimento para el buen funcionamiento del mecanismo, que sigue eliminando eficientemente el *HOL blocking* en cualquier circunstancia.

Estos resultados son incluso más importantes de lo que pudiera parecer en un principio, por las razones que se exponen a continuación. Nótese que las dos configuraciones de malla conectan el mismo número de terminales mediante un número distinto de conmutadores. En la malla de 4×4 conmutadores, la utilización de los enlaces será, en teoría, mayor que en la de 8×8, aumentando el riesgo de congestión. A su vez, en la malla 8×8 la longitud media de las rutas será mayor, aumentando el número de puntos donde pudiera aparecer *HOL blocking*. Pero, en ambos casos *RECN* soluciona estos problemas, ya que mantiene la productividad al máximo independientemente de la configuración de la malla. Por tanto, *RECN* permite conectar el mismo número de terminales con distintas configuraciones de la red sin riesgo a que aparezcan efectos "indeseables" debidos a los inconvenientes de una u otra configuración. Es decir: *RECN* aporta flexibilidad en cuanto al dimensionamiento de la red de interconexión.

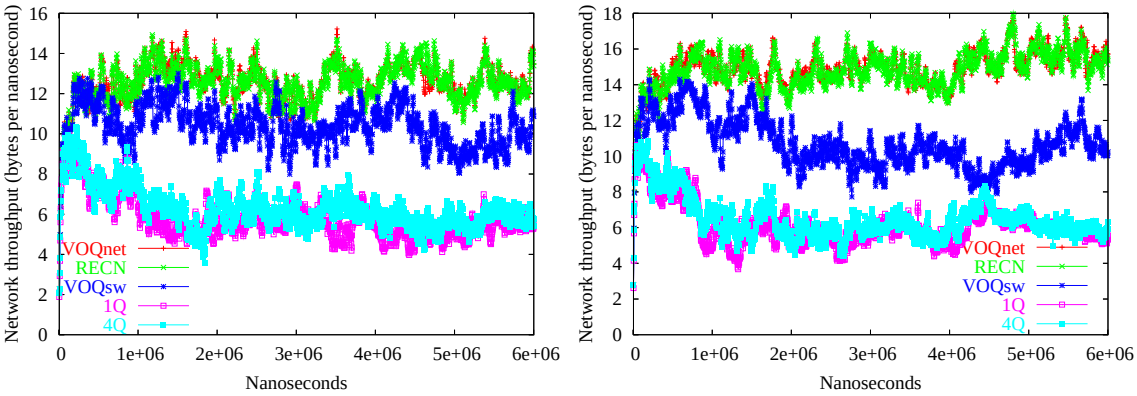
En la última figura del presente análisis (figura 5.35) se muestran resultados de productividad de la red en función del tiempo, obtenidos usando ficheros de trazas como carga de tráfico, con distintos factores de compresión de las mismas (factores 20 y 40, respectivamente). En todos los casos, la red de interconexión se ha simulado con una estructura $BMIN$ de 64 terminales. Las figuras 5.35.a y 5.35.b muestran resultados para conmutadores con valor de *speedup*=1,5, mientras que las figuras 5.35.c y 5.35.d contienen resultados para conmutadores de *speedup*=1. Siguiendo el formato habitual, cada gráfica individual incluye varias series de resultados correspondientes a distintas técnicas de eliminación del *HOL blocking*.

Como puede verse en la figura, de nuevo *RECN* y *VOQnet* consiguen las mejores prestaciones, con resultados prácticamente idénticos en todos los casos considerados. Esto significa que *RECN* es capaz de eliminar casi totalmente el *HOL* blocking también para cargas de tráfico más o menos "realista" (es decir, todo lo realista que puede considerarse un fichero de trazas). Las diferencias con los resultados obtenidos por el resto de técnicas siguen la línea observada en el análisis de figuras anteriores. Así, por ejemplo, el nivel de productividad obtenida para *VOQsw* se encuentra aproximadamente un 40 % por debajo del nivel obtenido por *RECN* para el caso de mayor diferencia (figura 5.35.d), que corresponde al tráfico con mayor factor de compresión y valor de *speedup*=1. Precisamente, si se comparan las figuras 5.35.d y 4.25.d (ambas muestran resultados para el mismo caso de tráfico y *speedup*, pero empleando la propuesta inicial y la mejorada, respectivamente) puede comprobarse de nuevo hasta qué punto mejora *RECN* con los cambios introducidos.

En resumen, habiendo mostrado resultados de productividad y latencia en función del tiempo y del tráfico generado, que han sido obtenidos para muy diversas cargas de



(a) FC 20, *speedup*=1,5 (b) FC 40, *speedup*=1,5



(c) FC 20, *speedup*=1 (d) FC 40, *speedup*=1

Figura 5.35: Productividad de la red en función del tiempo para las trazas con factores de compresión 20 (a y c) y 40 (b y d) y para las distintas técnicas de control de congestión consideradas. Conmutadores con *speedup*=1, 5 (a y b) y *speedup*=1 (c y d).

tráfico, para varias topologías de la red y para diferentes valores de *speedup*, puede afirmarse que, absolutamente en todos los casos, los resultados obtenidos usando *RECN* alcanzan el mejor nivel de los posibles. En cuanto a la productividad, *RECN* es capaz de mantenerla al máximo incluso con cargas altísimas de tráfico, independientemente de la topología de la red o el valor de *speedup* empleado. Respecto a la latencia, *RECN* es capaz de mantenerla a niveles mínimos, salvo para cargas de tráfico muy elevadas. Todo esto indica que la eficacia de *RECN* es prácticamente máxima como técnica de eliminación del *HOL blocking*, lo cual queda demostrado por el hecho de alcanzar siempre los mismos (si no mejores) resultados que *VOQnet*, la técnica teóricamente "ideal" en este sentido. Ahora bien, es esencial destacar que *RECN* consigue estas buenas prestaciones utilizando muchos menos recursos (el número máximo de *SAQs* por puerto es 8 para todos los casos) que *VOQnet*, que en la práctica es difícilmente implementable. Por el contrario, otras técnicas que también requieren pocos recursos (*VOQsw*, canales virtuales) no alcanzan buenos resultados, y se ven claramente superadas por *RECN*,

lo cual corrobora la idea de que, a diferencia de dichas técnicas, *RECN* sí elimina completamente el *HOL blocking*.

Por tanto, la eficiencia de *RECN* (de la versión mejorada, evidentemente) parece fuera de toda duda. Sin embargo, queda por demostrar que, además de eficiente, *RECN* es realmente escalable, manteniendo su eficacia en redes mucho mayores sin requerir más recursos. De no ser así, tendríamos una técnica con los mismos inconvenientes y ventajas que *VOQnet*, y nuestro trabajo no supondría un gran avance. De este asunto se ocupa plenamente el análisis siguiente.

5.3.5. Análisis de escalabilidad

El objetivo del presente análisis, como ya es sabido, es comprobar si la versión mejorada de *RECN* es capaz de alcanzar en redes de mayor tamaño las buenas prestaciones observadas en los análisis precedentes de este capítulo, sin que sea necesario para ello aumentar el número de recursos disponibles por el mecanismo (o sea, el número de *SAQs* por puerto).

Evidentemente, se trata de un análisis que debe basarse en resultados de *RECN* obtenidos para redes de tamaños mayores a las consideradas en análisis anteriores, y que podría desarrollarse, a diferencia del análisis anterior, al margen de los resultados obtenidos por otras técnicas (aunque sí se incluyen algunos resultados obtenidos para otras técnicas, a modo de referencia).

Por otra parte, nótese que del resultado de este análisis dependerá el que *RECN* cumpla plenamente o no con los objetivos marcados para la presente tesis en el sentido de proponer una técnica para la eliminación del *HOL blocking* que sea a la vez eficiente y escalable.

5.3.5.1. Configuración de las pruebas

Teniendo en cuenta el planteamiento del presente análisis, es obvio que las simulaciones correspondientes deben realizarse para redes de tamaños mayores que los considerados en los anteriores análisis de este capítulo. En concreto (y al igual que en el análisis de escalabilidad de la propuesta inicial), se han realizado pruebas para redes con topología $BMIN$ (*perfect shuffle*) de 512 y 2.048 terminales. Los resultados de dichas pruebas podrán de este modo compararse con los obtenidos para redes $BMIN$ de 64 terminales, que se han mostrado anteriormente. Se ha intentado que la única diferencia estructural entre las redes simuladas fuera el tamaño, para evitar introducir en el análisis excesivas variables que dificultaran la comparación de resultados. Este es el motivo por el que no se ha considerado oportuno variar la topología de la red.

Con la misma intención, la estructura de los computadores se ha mantenido idéntica, en la medida de lo posible, a la utilizada hasta ahora para $BMINs$: 8 puertos bidireccionales, valores de *speedup* de 1,5 ó 1 y memorias de 32 KB en cada entrada o salida. En cuanto a este último valor, existe una excepción (la única) a la norma de respetar la estructura de los conmutadores, ya que se han realizado algunas simulaciones con memorias de 128 KB en cada entrada o salida. El motivo para este cambio es que, para algunos casos de tráfico, se ha simulado, además del uso de *RECN*, el uso de *VOQnet*, que, al tratarse de redes de gran tamaño, requiere la existencia de un gran número de colas en cada puerto, para las cuales 32 KB no resultan suficientes. Por ejemplo, para redes con 512 terminales, y paquetes de 64 bytes, el tamaño de las 512 colas *VOQnet* en cada entrada o salida sería de 1 paquete, claramente insuficiente para evitar la inanición del *buffer* durante el envío de los créditos del control de flujo estándar. Con 128 KB las colas *VOQnet* tienen una capacidad de 4 paquetes de 64 bytes en redes de 512 terminales, lo cual no es demasiado, pero es suficiente. Para poder comparar resultados obtenidos en igualdad de condiciones, también se ha empleado un tamaño de 128 KB para las simulaciones de *RECN*, pero exclusivamente en aquellos casos de tráfico (los menos, por otra parte) en los que también se ha simulado *VOQnet*[15].

Por las mismas razones expuestas en los párrafos anteriores, para los *Input Adapters*, se mantienen las características empleadas hasta ahora, con la excepción indicada del tamaño de la memoria. Así, se han simulado en sus salidas 32 KB ó 128 KB (dependiendo del caso de tráfico), y como siempre, se asume división de los mensajes generados en paquetes de 64 bytes.

Respecto al tráfico, se han realizado simulaciones para todos los casos de tráfico sintético[16] mostrados en la tabla 5.3. Como puede comprobarse, existen patrones de tráfico ya empleados en el análisis de escalabilidad de la propuesta inicial (casos de tráfico 512-1, 512-2, 2048-1 y 2048-2) y nuevos patrones de tipo "incremental" (512-Inc-1 y 512-Inc-2). Los casos de tráfico ya empleados anteriormente sirven para ofrecer resultados en función del tiempo de simulación (con un intervalo concreto de inyección de tráfico congestionado), mientras que los nuevos patrones sirven para mostrar resultados en función de la carga variable de tráfico. Puede apreciarse que los tráficos incrementales siguen un esquema similar al resto, en cuanto a proporciones de fuentes enviando tráfico uniforme o congestionado, y en cuanto a que varía el número de árboles de congestión que se formarán en la red (1 ó 4, dependiendo del número de destinos preferentes en cada caso). Son precisamente estos casos incrementales con los que se ha simulado *VOQnet*, con las necesidades de memoria expuestas anteriormente. Es decir: el tamaño de la memoria de entradas y salidas es 32 KB en las simulaciones de todos

[15]En cualquier caso, los resultados que se obtienen en estos casos simulando *RECN* con 32 KB ó 128 KB son prácticamente idénticos.

[16]Cabe recordar que no disponemos de trazas par redes mayores de 64 terminales.

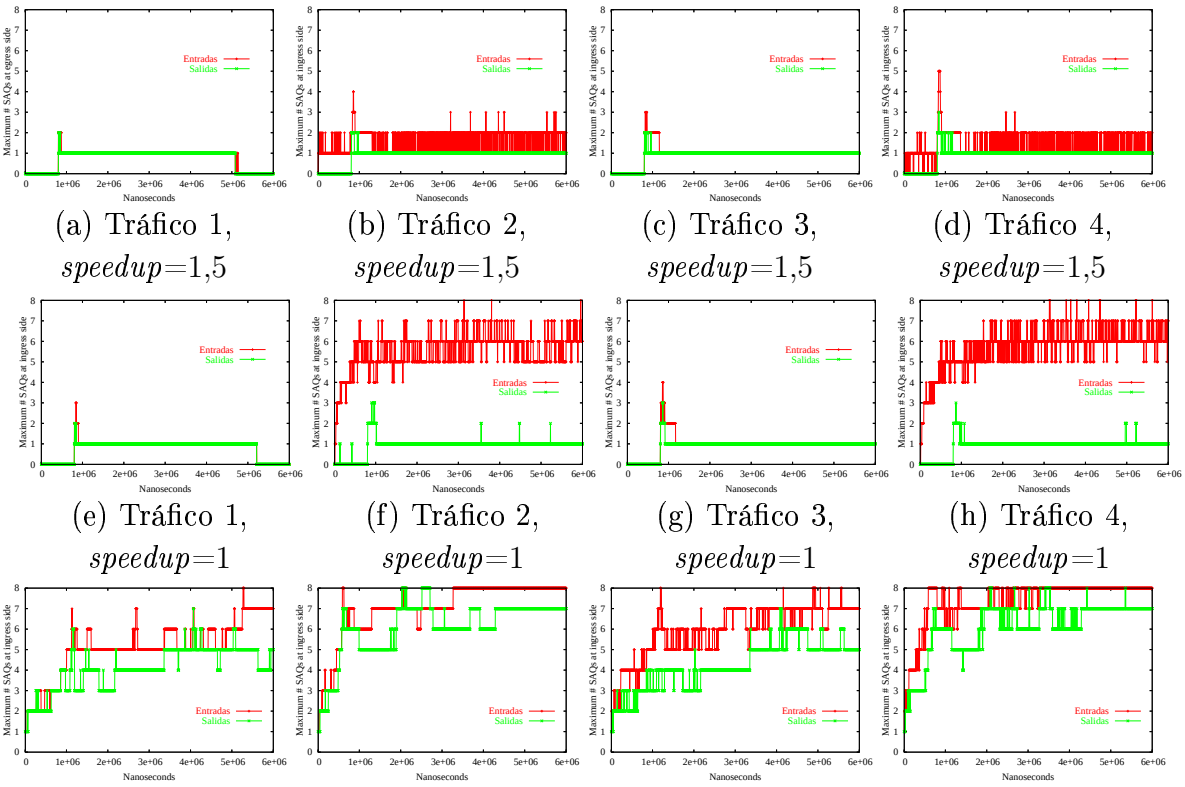
		Tráfico uniforme			Tráfico árbol de congestión				
Caso de tráfico	Tamaño de red (BMIN)	Nº. de fuentes	Destino	Tasa de generación	Nº. de fuentes	Destino	Tasa de generación	Instante inicio	Instante fin
512-1	512 × 512	75 %	aleatorio	100 %	25 %	Terminal 32	100 %	800 μs	1100 μs
512-2	512 × 512	75 %	aleatorio	100 %	6,25 %	Terminal 32	100 %	800 μs	1100 μs
					6,25 %	Terminal 201	100 %	800 μs	1100 μs
					6,25 %	Terminal 428	100 %	800 μs	1100 μs
					6,25 %	Terminal 500	100 %	800 μs	1100 μs
512-Inc-1	512 × 512	75 %	aleatorio	incremental	25 %	Terminal 32	incremental	0 μs	fin sim.
512-Inc-2	512 × 512	75 %	aleatorio	incremental	6,25 %	Terminal 32	incremental	0 μs	fin sim.
					6,25 %	Terminal 201	incremental	0 μs	fin sim.
					6,25 %	Terminal 428	incremental	0 μs	fin sim.
					6,25 %	Terminal 500	incremental	0 μs	fin sim.
2048-1	2048 × 2048	75 %	aleatorio	100 %	25 %	Terminal 32	100 %	800 μs	1100 μs
2048-2	2048 × 2048	75 %	aleatorio	100 %	6,25 %	Terminal 32	100 %	800 μs	1100 μs
					6,25 %	Terminal 515	100 %	800 μs	1100 μs
					6,25 %	Terminal 1540	100 %	800 μs	1100 μs
					6,25 %	Terminal 2000	100 %	800 μs	1100 μs

Tabla 5.3: Tráficos empleados en el análisis de escalabilidad para redes de 512 y 2.048 terminales.

los casos de tráfico, excepto los casos 512-Inc-1 y 512-Inc-2, para los cuales la memoria se asume de 128 KB.

En cuanto a las técnicas de eliminación del *HOL blocking*, se han realizado pruebas (evidentemente) con *RECN*, configurado con los valores de parámetros críticos establecidos en el análisis sobre el ajuste de los mismos: valores medios de los umbrales de detección y 8 *SAQs* como máximo por grupo. Nótese que un aspecto esencial del presente análisis es que este número máximo de *SAQs* se mantenga constante, ya que precisamente se pretende comprobar si esta cantidad de recursos sigue siendo suficiente para eliminar el *HOL blocking* en las redes de gran tamaño. También, y sólo como referencia de la intensidad de la congestión en la red, se han realizado simulaciones para *VOQsw*, con la configuración correspondiente a los conmutadores empleados (8 colas por puerto). Además, y sólo para los casos de tráfico incremental, como ya se ha comentado, se han realizado simulaciones para *VOQnet*, para tener una referencia de las prestaciones máximas alcanzable en la red. Por supuesto, el número de colas en entradas o salidas de *VOQnet* depende del número de terminales de la misma (512 o 2.048 colas, para las redes consideradas).

Por último, las métricas en las que se basa el análisis son por una parte, la productividad, para valorar hasta qué punto *RECN* es capaz de eliminar el *HOL blocking* en grandes redes, y por otra, el número máximo de *SAQs* activas en entradas o salidas, para valorar el consumo de recursos del mecanismo. Obviamente, la escalabilidad de *RECN* se valorará en la medida en que esta técnica elimine el *HOL blocking* en redes de distintos tamaños sin necesitar más recursos para ello.



(a) Tráfico 1, *speedup*=1,5 (b) Tráfico 2, *speedup*=1,5 (c) Tráfico 3, *speedup*=1,5 (d) Tráfico 4, *speedup*=1,5

(e) Tráfico 1, *speedup*=1 (f) Tráfico 2, *speedup*=1 (g) Tráfico 3, *speedup*=1 (h) Tráfico 4, *speedup*=1

(i) Trazas, FC=20, *speedup*=1,5 (j) Trazas, FC=40, *speedup*=1,5 (k) Trazas, FC=20, *speedup*=1 (l) Trazas, FC=40, *speedup*=1

Figura 5.36: Número máximo de *SAQs* activas en entradas y salidas en función del tiempo para los casos de tráfico sintético 1 a 4 (a-d para conmutadores con *speedup*=1,5, e-h para conmutadores con *speedup*=1) y para trazas con distinto factor de compresión (i-j para conmutadores con *speedup*=1,5, k-l para conmutadores con *speedup*=1).

5.3.5.2. Resultados

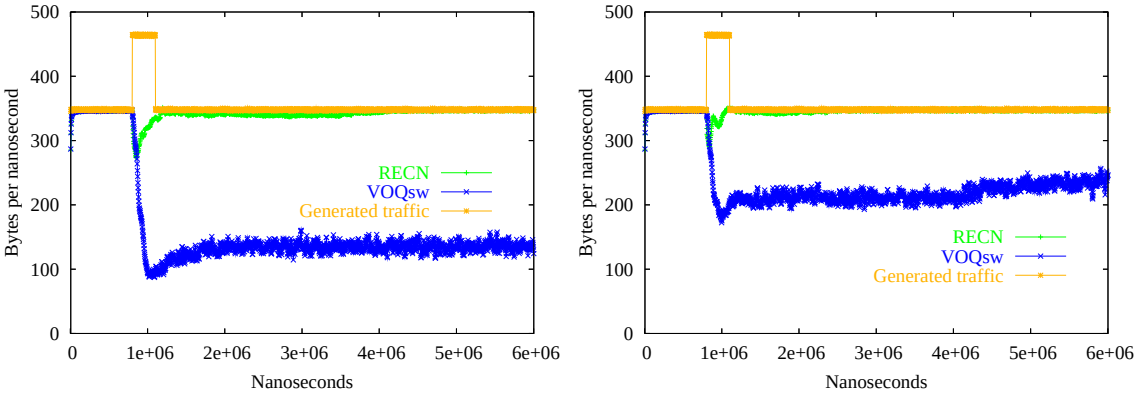
Aunque en el ajuste de los parámetros críticos de la nueva propuesta ya se han ofrecido resultados sobre el máximo número de *SAQs* activas para redes $BMIN$ de 64 terminales, conviene recuperar estos resultados para mostrarlos de una forma más clara (sólo para los valores de parámetros críticos seleccionados). Además, conviene valorar también el consumo de *SAQs* para otros tráficos no considerados en dicho ajuste.

Con esta intención, la figura 5.36 muestra el número máximo de *SAQs* activas en entradas y salidas durante el tiempo de simulación para los casos de tráfico 1 a 4 y para las trazas, en ambos casos para diferentes valores de *speedup* de los conmutadores (1,5 y 1, como es habitual). Cada gráfica individual contiene dos series que corresponden respectivamente a los resultados de consumo de *SAQs* en entradas (serie en rojo) y salidas (serie en verde), para un mismo caso de tráfico y un mismo valor de *speedup* (en el caso de las trazas, también para un mismo valor del factor de compresión).

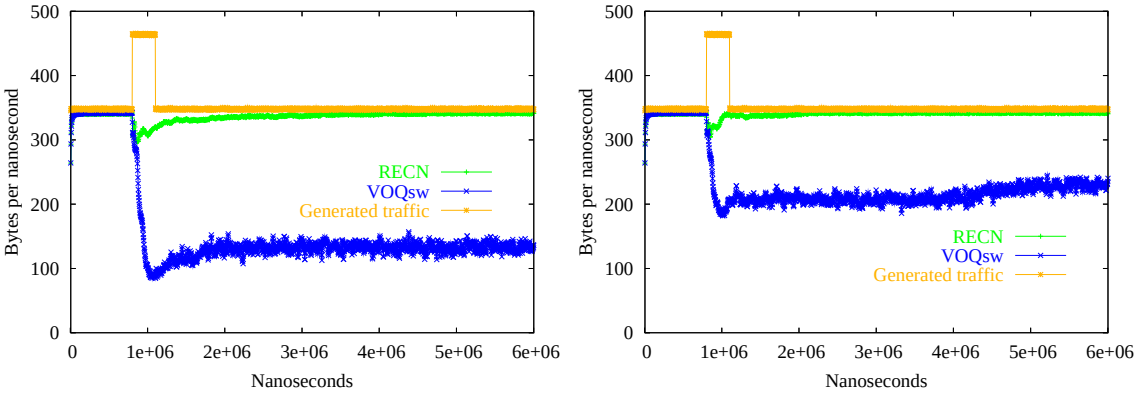
Como puede observarse, el consumo de *SAQs*, tanto en entradas como en salidas es normalmente inferior al máximo posible. El consumo es incluso muy inferior a 8 *SAQs* para varios de los casos considerados, especialmente en entradas y salidas para los casos de tráfico 1 a 4 en redes con valor de *speedup*=1,5 (figuras 5.36.a hasta 5.36.d). Para los casos de tráfico 1 a 4 en redes con valor de *speedup*=1 (figuras 5.36.e hasta 5.36.h), el consumo de *SAQs* es muy bajo en todas las salidas, y se acerca al máximo posible en las entradas, pero sólo para los casos de tráfico 2 y 4, más intensos que el resto (figuras 5.36.f y 5.36.h). El mayor consumo de *SAQs* en estas circunstancias se debe a usar detección de congestión en las entradas, junto con la mayor acumulación de paquetes en las mismas que se produce para redes con conmutadores sin aceleración interna. En cualquier caso, nótese que incluso en estas circunstancias sólo se llegan a usar 8 *SAQs* en muy pocos instantes. Respecto a los resultados de trazas (figuras 5.36.i hasta 5.36.l), el consumo de *SAQs* es especialmente alto para los casos con factor de compresión 40 (figuras 5.36.j y 5.36.l), donde, en las entradas, se activan todas las *SAQs* disponibles durante buena parte del tiempo. Evidentemente, este mayor consumo de *SAQs* está ligado a la alta tasa de generación de tráfico producida por el elevado factor de compresión. De todos modos, cabe recordar que, teniendo en cuenta los resultados del anterior análisis, *RECN* consigue que la productividad sea máxima incluso con esta exigente carga de tráfico.

Una vez claras las cifras del consumo de *SAQs* para las redes $BMIN$ de 64 terminales, el siguiente paso es comprobar si la versión mejorada de *RECN* elimina el *HOL blocking* eficientemente también en redes de gran tamaño. Para comenzar dicha comprobación, la figura 5.37 muestra resultados de productividad de la red en función del tiempo para una $BMIN$ de 512 terminales. Estos resultados se han obtenido para los casos de tráfico 512-1 y 512-2 (un destino *hot-spot* y cuatro destinos hot-spot, respectivamente), y para distintos valores de *speedup* de los conmutadores. Cada gráfica individual contiene tres series: dos de ellas corresponden a los resultados de *RECN* y *VOQsw* y la tercera (en amarillo) representa el tráfico generado a lo largo de la simulación. Esta última serie se muestra como referencia de la productividad máxima que puede alcanzarse. Los resultados para *VOQsw* se muestran como referencia de la intensidad de la congestión, que afectará a los resultados de dicha técnica al no eliminar ésta completamente el *HOL blocking*.

En la figura se aprecia claramente que, en todos los casos considerados, *RECN* consigue mantener la productividad prácticamente al nivel máximo (el marcado por el tráfico generado de forma uniforme, es decir, el nivel del tráfico generado antes y después del aumento producido en el intervalo de inyección de tráfico congestionado). Esto indica que, también para redes de este tamaño, *RECN* elimina eficientemente el *HOL blocking* que exista en al red. Más aún, teniendo en cuenta el nivel de degradación de la productividad que se aprecia en los resultados de *VOQsw* (una caída entre un 40 % y un 60 % respecto al nivel máximo, dependiendo del caso) el *HOL blocking* debe aparecer en la red con una intensidad considerable. Además, nótese que el buen



(a) Tráfico 512-1, *speedup*=1,5 (b) Tráfico 512-2, *speedup*=1,5

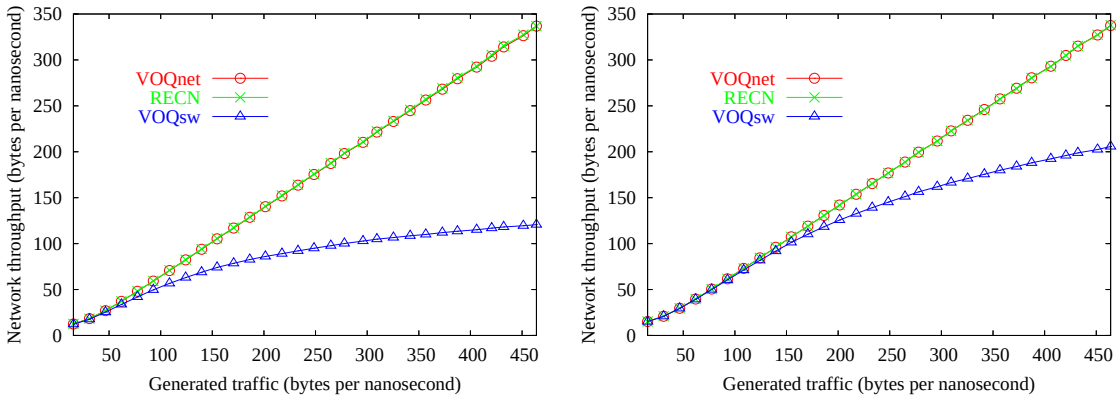


(c) Tráfico 512-1, *speedup*=1 (d) Tráfico 512-2, *speedup*=1

Figura 5.37: Productividad de la red en función del tiempo para los casos de tráfico sintético 512-1 y 512-2, y para conmutadores con *speedup*=1,5 y *speedup*=1.

comportamiento de RECN es independiente del valor de *speedup* de los conmutadores, o de que en la red aparezcan uno (tráfico 512-1) ó cuatro (tráfico 512-2) árboles de congestión. Téngase en cuenta que estos resultados son excelentes para *RECN*, pues se han conseguido sin aumentar el número máximo de *SAQs* por grupo (8), que sigue siendo el mismo empleado para redes con 64 terminales.

Para corroborar el buen comportamiento de *RECN* en redes grandes, la figura 5.38 muestra también resultados de productividad de la red para una $BMIN$ de 512 terminales, pero esta vez en función del tráfico generado. Para ser más precisos, se muestran los resultados para los casos de tráfico 512-Inc-1 y 512-Inc-2 (uno y cuatro destinos preferentes, respectivamente), y para valores de *speedup* de los conmutadores de 1,5 y 1. Las series contenidas en cada gráfica individual corresponden a los resultados para *RECN*, *VOQnet* y *VOQsw*. La serie para *VOQnet* se incluye esta vez como referencia de la productividad máxima, y la serie para *VOQsw* como referencia de la intensidad de la congestión. Recuérdese que al incluir resultados para *VOQnet*, ha sido necesario



(a) Tráfico 512-Inc-1, *speedup*=1,5

(b) Tráfico 512-Inc-2, *speedup*=1,5

(c) Tráfico 512-Inc-1, *speedup*=1

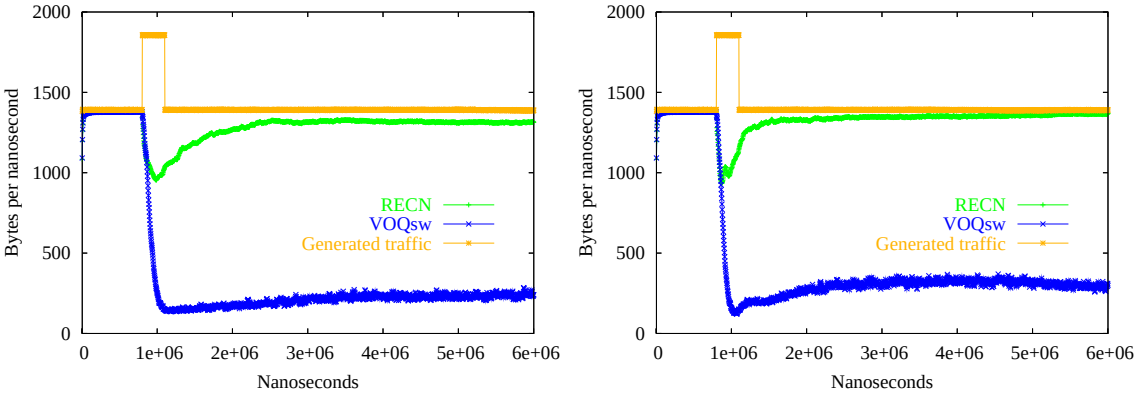
(d) Tráfico 512-Inc-2, *speedup*=1

Figura 5.38: Productividad de la red en función del tráfico generado para los casos de tráfico sintético 512-Inc-1 y 512-Inc-2, y para conmutadores con *speedup*=1,5 y *speedup*=1.

ajustar la memoria de los puertos a 128 KB en todas las simulaciones realizadas para obtener los resultados que se muestran en esta figura.

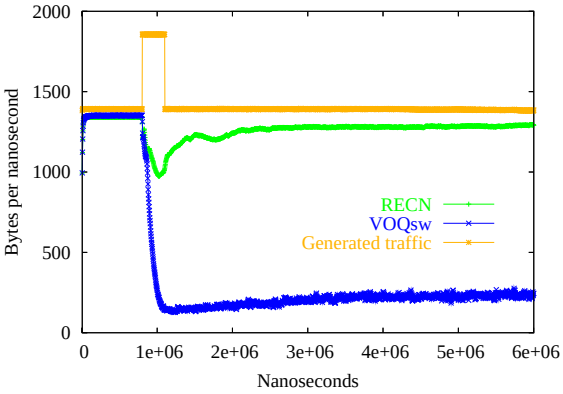
Siguiendo la tónica que se viene observando en los últimos análisis, la figura muestra que *RECN* consigue las mismas prestaciones que *VOQnet* en todos los casos considerados, lo que ratifica que el mecanismo mejorado elimina el *HOL blocking* de forma prácticamente total también en redes de gran tamaño. Igual que en la figura anterior, puede observarse que dicho *HOL blocking* no es en absoluto despreciable, si se considera la baja carga de tráfico para la que la red comienza a saturarse si se usa *VOQsw*. De nuevo, la relación entre los resultados de las distintas técnicas es independiente del valor de *speedup* de los conmutadores y de la presencia de más (tráfico 512-Inc-2) o menos (tráfico 512-Inc-1) árboles de congestión en la red.

Yendo más allá en cuanto al tamaño de la red, la figura 5.39 muestra resultados de productividad en función del tiempo para redes $BMIN$ de 2.048 terminales. En concreto, se muestran resultados obtenidos para los tráficos 2048-1 y 2048-2 (uno y cuatro

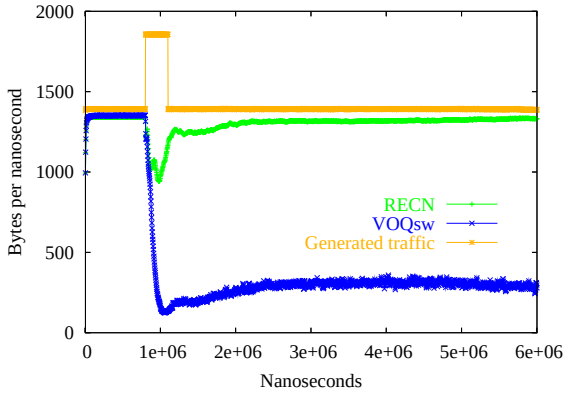


(a) Tráfico 2048-1, *speedup*=1,5

(b) Tráfico 2048-2, *speedup*=1,5



(c) Tráfico 2048-1, *speedup*=1

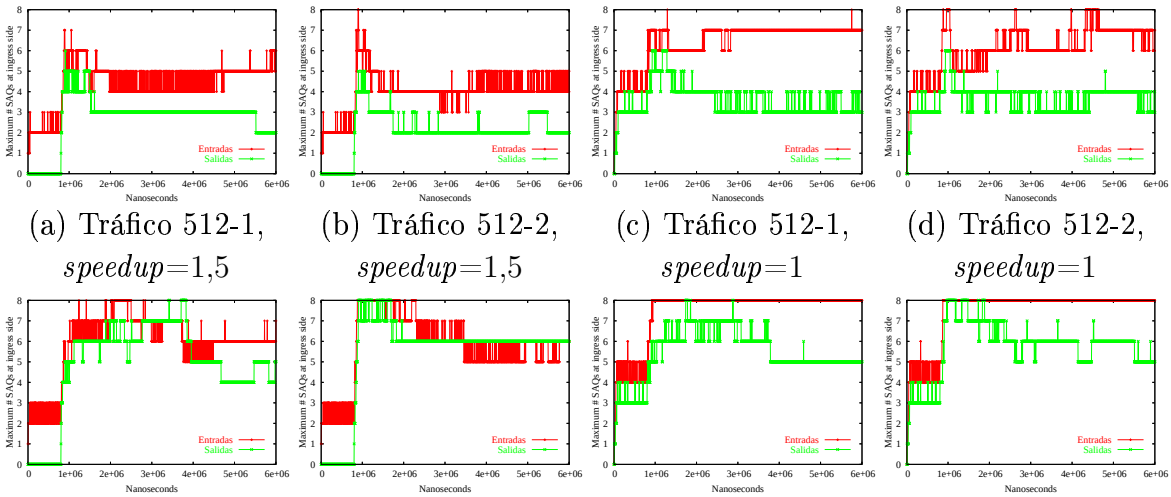


(d) Tráfico 2048-2, *speedup*=1

Figura 5.39: Productividad de la red en función del tiempo para los casos de tráfico sintético 2048-1 y 2048-2, y para conmutadores con *speedup*=1,5 y *speedup*=1.

destinos preferentes, respectivamente) y para valores de *speedup* de los conmutadores de 1,5 y 1. Como es habitual, cada gráfica individual contiene tres series: una serie con los resultados de *RECN*, una con los resultados de *VOQsw* (como referencia de la intensidad de la congestión) y otra que representa el tráfico generado en cada instante.

La figura muestra de nuevo buenos resultados de *RECN*, aunque se observa un periodo transitorio, que comienza coincidiendo con la inyección de tráfico congestionado, en el que la productividad desciende (hasta un 20 % respecto al máximo) para luego recuperarse. Esta bajada de productividad durante el transitorio es debida tanto a la enorme intensidad de la congestión de los casos planteados (téngase en cuenta que esta intensidad es tal que la productividad obtenida usando *VOQsw* llega a caer hasta un 90 %) como al hecho de tratarse de redes con un gran número de etapas, donde las ramas de los árboles de congestión tendrán una longitud mucho mayor. Por tanto, el mecanismo tarda un poco en asignar *SAQs* en todos los puntos por donde pasen las ramas del árbol, y es en estos puntos donde se produce *HOL blocking* durante algún tiempo. De hecho, este transitorio también aparece en los resultados para redes de



(a) Tráfico 512-1, *speedup*=1,5 (b) Tráfico 512-2, *speedup*=1,5 (c) Tráfico 512-1, *speedup*=1 (d) Tráfico 512-2, *speedup*=1

(e) Tráfico 2048-1, *speedup*=1,5 (f) Tráfico 2048-2, *speedup*=1,5 (g) Tráfico 2048-1, *speedup*=1 (h) Tráfico 2048-2, *speedup*=1

Figura 5.40: Número máximo de *SAQs* activas en entradas y salidas en función del tiempo para los casos de tráfico sintético 512-1 y 512-2 (a y b para conmutadores con *speedup*=1,5, c y d para conmutadores con *speedup*=1) y para los casos 2048-1 y 2048-2 (e y f para conmutadores con *speedup*=1,5, g y h para conmutadores con *speedup*=1).

menor tamaño, aunque de forma mucho menos apreciable. Ahora bien, una vez pasado el transitorio, *RECN* habrá asignado todas las *SAQs* necesarias para eliminar el *HOL blocking* producido por los paquetes que forman el árbol de congestión, y gracias a ello es capaz en todos los casos de mantener la productividad prácticamente al máximo nivel, incluso en estas condiciones de tráfico tan intenso. Nótese la gran diferencia con los resultados de *VOQsw*, donde la productividad no sólo se desploma, sino que además no se recupera.

En suma, y a la vista de los resultados de productividad para redes de 512 y 2.048 terminales, se demuestra que *RECN* es capaz de eliminar, también en redes de gran (o muy gran) tamaño, el *HOL blocking* que pudiera aparecer, incluso en condiciones muy adversas en cuanto a la intensidad de la congestión a resolver. Y, lo más importante, *RECN* consigue esto sin incrementar el número máximo de *SAQs* por grupo que se fijó para redes menores.

Ahora bien, este número fija la cantidad de *SAQs* disponibles en cada puerto, que no siempre estarán activas, y por ello puede resultar de interés comprobar los niveles de consumo "real" de *SAQs* en redes de gran tamaño. Con esta intención, la figura 5.40 muestra el número máximo de *SAQs* activas en entradas y salidas en función del tiempo, para los casos de tráfico 512-1, 512-2, 2048-1 y 2048-2, y para valores de *speedup* de los conmutadores de 1,5 y 1. Esta figura se ajusta a un esquema similar al de la figura 5.36, que muestra la misma información para redes de 64 terminales.

Estos resultados confirman no sólo que 8 *SAQs* por grupo son suficientes para eliminar el *HOL* blocking en redes de gran tamaño, sino también que en muchos casos ni siquiera llegan a utilizarse todas estas *SAQs* disponibles. Sólo para redes de 2.048 terminales y conmutadores de *speedup*=1 (figuras 5.40.g y 5.40.h) se observa una activación prolongada del máximo número de *SAQs* posible. Ahora bien, si comparamos ambas figuras con la figura 5.36.l, que muestra resultados para una $BMIN$ de 64 terminales, se observará que todas ellas son bastante similares. Esto indica que el consumo de *SAQs* no depende tanto del tamaño de la red como de otros factores, como la intensidad del tráfico o el valor de *speedup*. Respecto a este último, parece un factor muy importante en cuanto al consumo de *SAQs*, ya que cuando es 1 se consumen muchas más *SAQs* en las entradas debido, como ya se ha comentado, a la mayor acumulación de paquetes en las mismas, y a las correspondientes detección y asignación de *SAQs*. En cualquier caso, es evidente que el hecho de que todas las *SAQs* disponibles estén activas en alguna entrada no supone ningún problema para que el mecanismo consiga una altísima eficiencia eliminando el *HOL blocking*.

En definitiva, a partir de los resultados expuestos en esta sección y en la anterior, podemos afirmar que *RECN*, tras las mejoras introducidas y detalladas en este capítulo, es capaz de eliminar el *HOL blocking* de forma prácticamente total en redes de cualquier tamaño, y requiriendo para ello una cantidad reducida y constante de recursos (8 *SAQs* por grupo). Esto quiere decir que el mecanismo, en su versión final, cumple plenamente con los objetivos marcados para nuestra investigación, en el sentido de desarrollar una técnica de eliminación del *HOL blocking* eficiente y escalable.

5.3.6. Análisis adicionales

Aunque la valía de *RECN* ha quedado ya plenamente demostrada en los análisis anteriores, hemos considerado oportuno ofrecer más datos sobre el mecanismo, principalmente de cara a aclarar algunas cuestiones que pudieran plantearse sobre su implementación real.

En este sentido, los dos siguientes puntos se ocupan, respectivamente, de analizar la sobrecarga que pudieran introducir los paquetes de control empleados por el mecanismo, y de estudiar (de forma aproximada) las necesidades en cuanto a área de silicio de las estructuras de memoria de datos establecidas por *RECN*. Téngase en cuenta que, al tratarse de análisis complementarios de los anteriores, la extensión y profundidad de las explicaciones que se incluyen en los siguientes puntos serán menores que las habituales.

5.3.6.1. Sobrecarga debida a paquetes de control

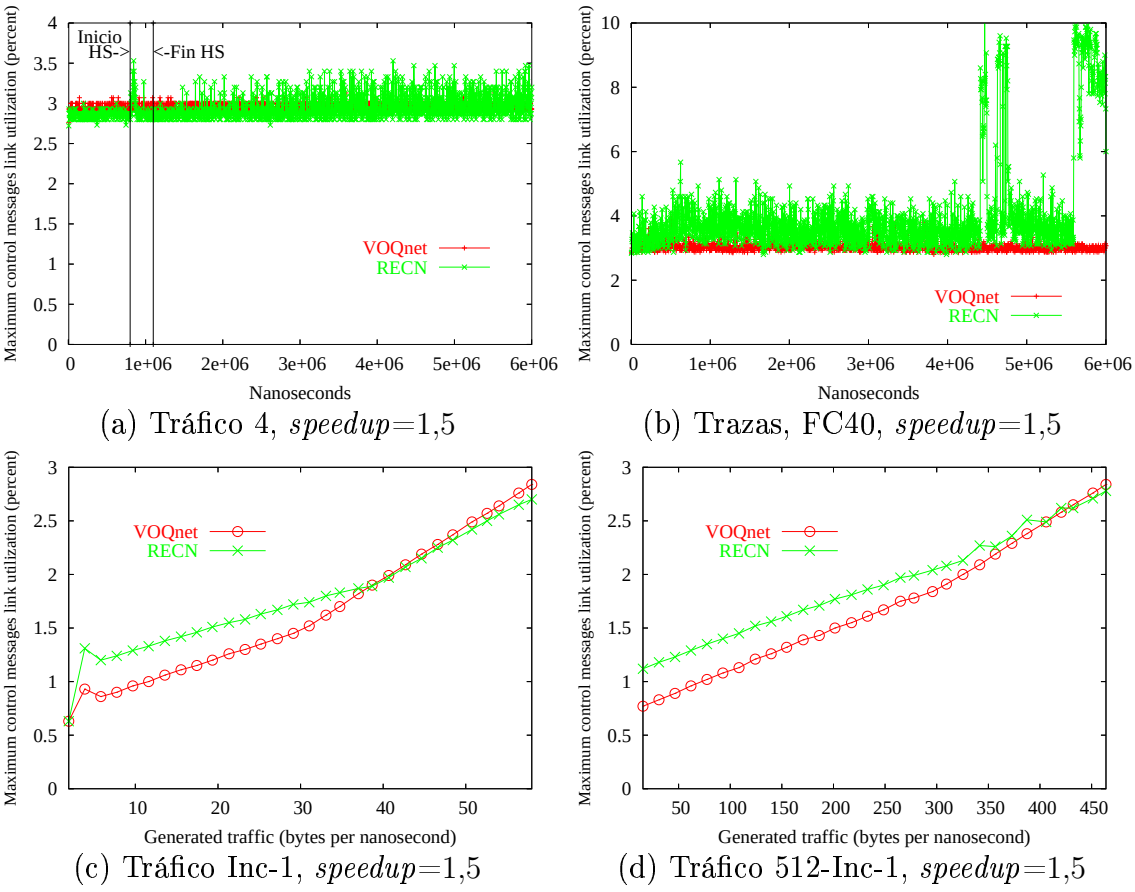
A pesar de que uno de los cambios introducidos en *RECN* en este capítulo es la eliminación de las notificaciones de liberación de una *SAQ*, lo cierto es que el funcionamiento del mecanismo sigue requiriendo la transmisión de ciertos mensajes de control entre puertos de los conmutadores. Dichos mensajes siguen siendo necesarios, concretamente, para los mecanismos de control de flujo *Xon/Xoff* y de propagación de información de congestión.

Evidentemente, estos mensajes de control no se transmitirían si se usase otra técnica de control de congestión (o si no se usase ninguna), por lo que cabría plantear si el uso de *RECN* sobrecarga los enlaces por culpa de este trasiego "extra" de información. Evidentemente, hay que tener en cuenta que existirán otros mensajes de control imprescindibles para el funcionamiento de la red, y que no son "exclusivos" de *RECN* (como los propios del control de flujo "convencional"), por lo que no puede culparse a *RECN* de toda la carga de mensajes de control en los enlaces de la red.

De cara a esclarecer este punto, hemos obtenido resultados que muestran la máxima utilización de un enlace por mensajes de control, midiendo el porcentaje del tiempo dedicado a transmitir mensajes de control por cada enlace de la red, y tomando el resultado del enlace para el que este porcentaje es mayor. Estos resultados se han obtenido para algunos de los casos de tráfico y dimensiones de la red utilizados en análisis anteriores, procurando seleccionar configuraciones variadas que permitieran extraer unas conclusiones lo más generales posibles.

Para poder evaluar la sobrecarga introducida por *RECN*, hemos considerado los resultados de simulaciones realizadas asumiendo el uso de esta técnica, pero también se han tomado resultados de simulaciones para *VOQnet*, cuyo uso no implica, en principio, ninguna mensajería especial. En las simulaciones para *RECN* se han considerado mensajes de control los correspondientes al control de flujo convencional (el basado en créditos), al control de flujo *Xon/Xoff*, y a la propagación de la información de congestión. En el caso de simular *VOQnet* se han considerado mensajes de control únicamente los correspondientes al control de flujo basado en créditos. De este modo, la comparación de los resultados para ambas técnicas permitirá evaluar la "sobrecarga" realmente introducida por *RECN*.

La figura 5.41 muestra los resultados obtenidos. En las figuras 5.41.a y 5.41.b el máximo porcentaje de uso del enlace por mensajes de control se muestra como una función del tiempo, a lo largo de una simulación. En ambos casos se trata de resultados para una $BMIN$ de 64 terminales, usando como carga bien el tráfico sintético 1 (figura 5.41.a), bien trazas con factor de compresión 40 (figura 5.41.b). Las figuras 5.41.c y 5.41.d, en cambio, muestran los resultados en función de la carga de tráfico. En ambos casos se ha usado como carga tráfico sintético (tráficos Inc-1 y 512-Inc-1, respectivamente), pero en redes $BMIN$ de distinto tamaño: 64 terminales (figura 5.41.c) y 512



(a) Tráfico 4, *speedup*=1,5

(b) Trazas, FC40, *speedup*=1,5

(c) Tráfico Inc-1, *speedup*=1,5

(d) Tráfico 512-Inc-1, *speedup*=1,5

Figura 5.41: Máxima utilización de un enlace (%) de la red por paquetes de control para distintos casos de tráfico.

terminales (figura 5.41.d). Obviamente, cada gráfica individual muestra dos series: una para los resultados de *RECN* (en verde) y otra para los resultados de *VOQnet* (en rojo).

De los resultados expuestos se deduce que, efectivamente, *RECN* introduce cierto grado de sobrecarga de mensajes de control en los enlaces, pero dicha sobrecarga no es en absoluto excesiva. En la mayoría de las configuraciones evaluadas, la diferencia máxima entre los resultados de *RECN* y *VOQnet* es aproximadamente de un 0,5 % de utilización del enlace. Sólo para el caso de uso de trazas como carga de tráfico (figura 5.41.c) aparecen "picos" donde los resultados de *RECN* superan a los de *VOQnet* en un 7 % de utilización del enlace. En la misma configuración, y fuera de estos picos, la diferencia entre ambas técnicas es mucho menor. Dichos picos pueden deberse a fluctuaciones extremas del tráfico (varias "ráfagas" a intervalos en las trazas, por ejemplo), donde, en intervalos cortos de tiempo, se asignen, se liberen y vuelvan a asignarse *SAQs*, que además probablemente pasarán por los estados *Xon* y *Xoff* repetidas veces. De todos modos, incluso en estos casos la sobrecarga no puede considerarse exagerada, y

más teniendo en cuenta que los resultados se muestran para el enlace más sobrecargado de mensajes de control de toda la red.

En cualquier caso, queda comprobado que la sobrecarga debida a los mensajes de control empleados por *RECN* no es excesiva. Y, por supuesto, teniendo en cuenta los resultados de análisis anteriores, dicha sobrecarga no supone ningún impedimento para conseguir excelentes prestaciones de la red cuando se usa *RECN* como técnica de eliminación del *HOL blocking*.

5.3.6.2. Requisitos de área de silicio

Tal y como se ha repetido en numerosas ocasiones a lo largo de esta memoria de tesis, uno de los principales motivos para proponer *RECN* es la incompatibilidad entre escalabilidad y eficiencia que se observaba en las técnicas propuestas anteriormente para la eliminación del *HOL blocking*.

En concreto, la técnica teóricamente más eficiente era *VOQnet*, que como es sabido necesita una cantidad de colas en cada puerto igual al número de destinos en la red, por lo que no es escalable respecto al tamaño de la misma. La consecuencia práctica más evidente de esta falta de escalabilidad, es, naturalmente, que la implementación real de todas las colas necesarias en redes grandes sería muy costosa, ya que para cada cola se requiere un tamaño mínimo del área de silicio dedicada a memoria de datos, y, como es obvio, a mayor área de silicio, mayor coste de los componentes de red.

Como también se ha repetido, *RECN* consigue las mismas prestaciones que *VOQnet* con un número más reducido y constante de recursos. Concretamente, en el peor de los casos (en las entradas) *RECN* necesita tantas colas de detección como salidas tenga el conmutador, más tantas *SAQs* como se permitan por grupo. En general, y teniendo en cuenta que (según lo expuesto anteriormente) a *RECN* le bastan 8 *SAQs* para eliminar eficazmente el *HOL blocking*, el número de colas necesarias para *RECN* en cada puerto será muy inferior al número requerido por *VOQnet*. Esto evidentemente supone, de cara a una posible implementación, un considerable ahorro en el área de silicio para memoria de datos.

Un estudio exacto del área de silicio mínima necesaria para implementar realmente estas colas no es una tarea inmediata, y se encuentra fuera de los objetivos de la presente tesis. Ahora bien, es posible ofrecer una estimación (aproximada, obviamente) de la magnitud del ahorro en área de silicio que *RECN* puede aportar.

Para ello, en primer lugar, la figura 5.42 muestra el total de colas requeridas por distintas técnicas de eliminación del *HOL blocking* en cada conmutador, para redes de distinto tamaño y para conmutadores con distinto número de puertos. Como puede verse, se muestran datos para *VOQnet*, *VOQsw* y *RECN*, asumiendo en este último

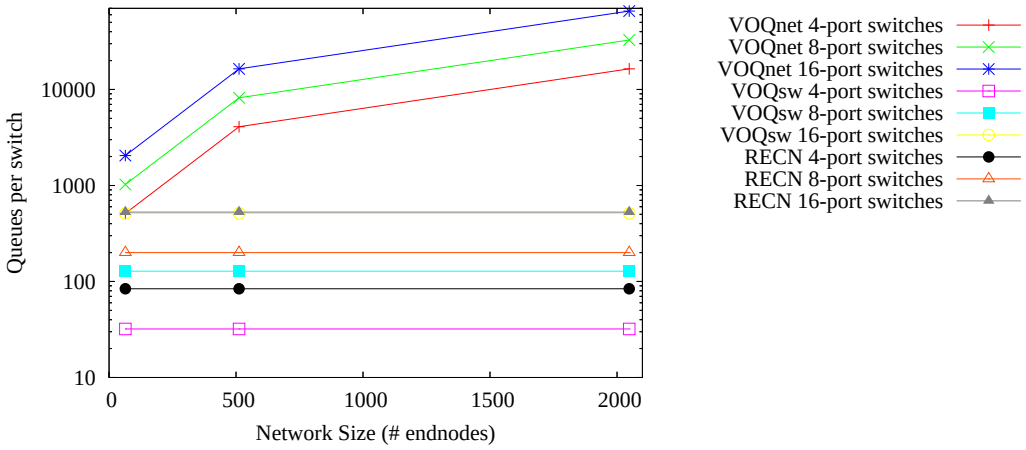


Figura 5.42: Número total de colas necesarias en un conmutador, en función del número de puertos del mismo y de la técnica de eliminación del *HOL blocking* empleada.

caso 8 *SAQs* por grupo. Téngase en cuenta en la figura el uso de la escala logarítmica en el eje correspondiente al número de colas.

La figura muestra claramente las enormes diferencias que pueden llegar a darse entre las colas requeridas por *VOQnet* y las requeridas por técnicas escalables (*VOQsw* y *RECN*), cuyas necesidades no aumentan con el tamaño de la red. Naturalmente, la descomunal cantidad de colas requeridas por *VOQnet* en redes de tamaño medio o alto hace impensable su implementación. Cabe destacar que el número de colas necesario por *RECN* es siempre ligeramente superior al requerido por *VOQsw*. Ahora bien, esta pequeña diferencia entre ambas técnicas en cuanto a requisitos queda eclipsada por la gran diferencia a favor de *RECN* en cuanto a prestaciones.

Obviamente, las diferencias observadas en cuanto al número de colas necesarias tendrán un reflejo en el área de silicio requerida para la implementación de dichas colas. En este sentido, existen tablas (*Memory Datasheets*) que permiten obtener un valor aproximado del área de silicio requerida para implementar una cantidad de memoria (en *bytes*), en función de la tecnología empleada.

Tras consultar estas tablas, hemos obtenido una estimación del área de silicio para memoria de datos necesaria para implementar las colas requeridas en puertos de entrada o salida por las distintas técnicas en algunas configuraciones de red. Concretamente, para seleccionar el *Datasheet* hemos asumido el uso de memoria *SRAM* de tecnología 0,18 μm *CMOS* con una organización de 32 bits. En cuanto a los *bytes* mínimos necesarios para cada cola, hemos asumido que cada cola debe tener espacio para, al menos, el tamaño máximo de un paquete *Advanced Switching*: 2 KB. En cuanto a las configuraciones de red, hemos realizado cálculos para conmutadores de 8 puertos, en redes

	Área de silicio para memoria de datos			
	Red de 64 terminales		Red de 512 terminales	
Técnica de Control de congestión	Entradas	Salidas	Entradas	Salidas
VOQnet	$16mm^2$	$16mm^2$	$128mm^2$	$128mm^2$
RECN (8 SAQs)	$4mm^2$	$2{,}25mm^2$	$4mm^2$	$2{,}25mm^2$
VOQsw	$2mm^2$	$2mm^2$	$2mm^2$	$2mm^2$

Tabla 5.4: Área de silicio requerida en cada puerto de entrada o salida para distintas técnicas de control de congestión.

de 64 y 512 terminales. Para *RECN* se han asumido 8 *SAQs* por grupo. Los valores estimados se muestran en la tabla 5.4.

Como no podía ser de otro modo, puede comprobarse en la tabla que *VOQnet* requiere un área de silicio por puerto de entrada o salida mucho mayor que la necesaria para implementar las otras técnicas. De hecho, las diferencias en el caso de redes grandes son de dos órdenes de magnitud, lo cual es sumamente elocuente. Por otra parte, y también como era de suponer, *RECN* requiere algo más de área de silicio por puerto que *VOQsw*, pero realmente las diferencias son sólo significativas en las entradas. Además, el área de silicio requerida por *RECN* en las entradas no parece excesivo en ningún caso, y el ahorro en silicio que supondría el uso de *VOQsw* difícilmente compensaría la pérdida de prestaciones que sufriría la red con esta técnica.

En resumen, aunque desde el punto de vista teórico ya era evidente que *RECN* requería muchos menos recursos que *VOQnet*, las estimaciones sobre el área de silicio mostradas en este punto nos permiten ofrecer cifras "palpables" para valorar hasta qué punto el empleo de *RECN* supone un ahorro cuando se desea contar con una técnica que elimine totalmente el *HOL blocking*.

5.4. Conclusiones

En el presente capítulo se han expuesto, en primer lugar, los defectos de la propuesta inicial de *RECN*, responsables de que su eficiencia resultara a todas luces insuficiente, tal y como pudo comprobarse en el capítulo anterior. La detección de estos puntos débiles del mecanismo fue posible gracias al estudio sobre la dinámica de los árboles de congestión que también se muestra en este capítulo. De hecho, como se ha explicado, el origen de dichos defectos es, en general, el no haberse considerado en la propuesta inicial todas las posibles formas de aparición, crecimiento y desaparición de los árboles de congestión.

La identificación precisa de los problemas de *RECN* nos permitió a su vez buscar formas concretas de modificarlo de tal manera que pudieran resolverse todos los

inconvenientes. De este modo surgieron los cambios en que se han detallado convenientemente en la sección 5.2, y que mejoran la propuesta inicial para que, en general, *RECN* se adapte a cualquier tipo de formación o desvanecimiento de los árboles de congestión, sin que ningún caso especial en este sentido impida el funcionamiento óptimo del mecanismo.

Teniendo en cuenta los resultados de la evaluación de la propuesta mejorada, es evidente que no nos equivocamos ni en el diagnóstico de los problemas que aquejaban a la propuesta inicial, ni en el remedio propuesto para solventarlos. Como hemos visto, la versión mejorada de *RECN* consigue excelentes resultados para todos los casos considerados en los distintos análisis realizados para comprobar su validez que se muestran en el presente capítulo. Creemos que dichos análisis, por su parte, son lo suficientemente amplios, variados y precisos como para estar seguros de las conclusiones que de ellos hemos extraído.

Concretamente, hemos comprobado que la nueva versión de *RECN* permite obtener de la red la máxima productividad (al mismo nivel que *VOQnet*, teóricamente ideal) independientemente del tamaño o topología de la misma, incluso en situaciones de congestión intensa. Además, estos buenos resultados de productividad se consiguen también para conmutadores sin aceleración interna, a diferencia de lo que sucedía con la propuesta inicial. *RECN* consigue, además, mantener la latencia de red de los mensajes al menor nivel de los posibles, incluso por debajo de los valores obtenidos por *VOQnet*. Todos estos resultados indican que *RECN* elimina de forma prácticamente total el *HOL blocking* que pueda aparecer en la red, en cualquier circunstancia.

Pero la auténtica relevancia de estos buenos resultados de *RECN* es que se han conseguido, tal y como se pretendía, requiriendo una cantidad modesta y constante (idéntica para todas las configuraciones y tamaños de red) de recursos, en cuanto a memoria se refiere. De este modo, *RECN* consigue unas prestaciones del mismo nivel (o mayor) que las conseguidas por *VOQnet* (técnica eficiente, difícilmente implementable) con unos requisitos en cuanto a memoria similares a los de *VOQsw* (técnica implementable, pero poco eficaz). Es decir, la escalabilidad de *RECN*, tras los cambios introducidos, también ha quedado demostrada en este capítulo.

Por todo ello, consideramos que esta versión mejorada de *RECN* cumple plenamente con el objetivo principal de nuestra investigación, ya que hemos obtenido una técnica totalmente eficiente y escalable para la eliminación del *HOL blocking*.

Capítulo 6

Conclusiones, aportaciones y trabajo futuro

En los capítulos precedentes se han detallado la justificación, planteamiento y desarrollo de todos los estudios y trabajos enmarcados en la presente tesis. En este capítulo, el último de la memoria de tesis, se ofrece una recapitulación final sobre la investigación realizada, indicando tanto las principales aportaciones de la misma como las conclusiones más importantes que de ella pueden extraerse. También se incluye una lista de las publicaciones a las que esta investigación ha dado lugar, así como algunos datos de diversos proyectos que han contribuido a financiarla. Por último, se apuntan algunas ideas sobre futuras líneas de investigación que podrían plantearse como continuación de la presentada en esta memoria.

6.1. Aportaciones

En la sección 1.4 se plantearon los objetivos específicos de la presente tesis, de modo que, obviamente, las aportaciones que ésta ofrece vienen determinadas por aquellos de estos objetivos que puedan considerarse cumplidos. Teniendo esto en cuenta, las principales aportaciones de la tesis se enumeran y resumen a continuación, indicando en cada caso con cuál o cuáles de los mencionados objetivos existe una relación directa:

1. **Se ha realizado un profundo análisis del fenómeno de la congestión en redes de interconexión**. Este análisis ha ido más allá de las habituales reflexiones acerca de las causas y consecuencias inmediatas de las situaciones de congestión, ya que también se han caracterizado diferentes tipos de evolución de los árboles de congestión, en cuanto a su crecimiento y su desaparición, y se han detectado los principales factores que condicionan estos procesos. Las conclusiones de este análisis han sido fundamentales para conseguir las mejores prestaciones de

nuestra propuesta, que fue adaptada para responder eficientemente ante cualquier forma de evolución de los árboles de congestión. Por otra parte, y dado que no tenemos noticias de que existieran estudios anteriores en este sentido, creemos que esta aportación puede ser de gran utilidad para el desarrollo de futuras técnicas relacionadas con la congestión en redes de interconexión. Evidentemente, esta aportación supone cumplir con creces el primero de los objetivos planteados en la sección 1.4.

2. **Se han identificado las ventajas e inconvenientes de las distintas propuestas anteriores para el control de congestión en redes de interconexión**. Para ello, se han considerado tanto las características y necesidades de las actuales redes de interconexión como los distintos enfoques con los que se ha abordado tradicionalmente el problema de la congestión. En general, buena parte de los problemas que afectan a estas propuestas se deben a un enfoque demasiado "estático" respecto a la forma de combatir la congestión, que lleva bien a consumir demasiados recursos (falta de escalabilidad), bien a utilizar los existentes de forma inapropiada (falta de eficacia). El conocimiento de dichas ventajas e inconvenientes ha sido esencial para justificar y orientar correctamente nuestra propuesta, del mismo modo que puede ser útil para evitar puntos débiles en futuras técnicas de control de congestión. Como puede comprobarse, esta aportación se corresponde con el segundo objetivo de la tesis, según la lista presentada en la sección 1.4.

3. **Se ha propuesto una nueva técnica para el control de congestión en redes de interconexión**. Dicha técnica ha sido llamada *RECN* (*Regional Explicit Congestion Notification*), y es la primera que ofrece a la vez máxima eficiencia en cuanto al control de congestión y plena escalabilidad, ya que, requiriendo una cantidad reducida y constante de recursos, es capaz de eliminar totalmente el *HOL blocking*, principal causa de la degradación de prestaciones de la red en situaciones de congestión. *RECN* es aplicable en redes de cualquier tecnología, independientemente de su topología o tamaño, siempre que se permita el uso de encaminamiento fuente determinista. Como se ha explicado repetidamente, *RECN* se basa en detectar la aparición de congestión en cualquier punto de la red (bien en el enlace de acceso a un terminal, bien en el interior de la red), por lo que localiza la raíz de los árboles de congestión con toda exactitud, independientemente de donde se sitúe ésta. Esto permite a su vez identificar de forma precisa los paquetes que pertenecen a estos árboles de congestión, que son almacenados en colas especiales (*SAQs*), asignadas de forma dinámica. Esta gestión dinámica de los recursos permite al mecanismo, por una parte, adaptarse a las complejas y variables formas de evolución de los árboles de congestión, y por otra, utilizar los recursos disponibles sólo cuando es estrictamente necesario. Puesto que los paquetes no congestionados se almacenan en colas distintas a las *SAQs*, se evita que compartan cola con los paquetes congestionados, lo que elimina el *HOL*

blocking que estos últimos pudieran producir. Los paquetes no congestionados no producen *HOL blocking* relevante, y por tanto pueden mezclarse en la misma cola sin peligro, lo que redunda en un menor consumo de recursos. Por todo lo expuesto, *RECN* aporta a las actuales redes de interconexión la posibilidad de trabajar cerca del punto de saturación sin riesgo de sufrir la degradación de prestaciones que producen las situaciones de congestión, a costa de implementar una cantidad moderada de recursos en cada puerto. Esta aportación permite por ejemplo, obtener las máximas prestaciones de la red en condiciones de alta carga de tráfico, o también conectar un mayor número de terminales por conmutador sin riesgo a que la mayor utilización de los enlaces ocasione problemas. Nótese que esta última posibilidad implica un ahorro notable en cuanto al coste de la red y en cuanto al consumo de potencia en la misma. Obviamente, esta aportación satisface el tercero de los objetivos marcados para la tesis en la sección 1.4.

4. **Se ha evaluado y optimizado la técnica propuesta**. Para ello, se ha recurrido a un simulador desarrollado expresamente para esta evaluación, que ofrece métricas representativas de las prestaciones de la red (productividad, latencia, etc.) en condiciones configurables por el usuario. Con esta herramienta se ha realizado una amplia gama de simulaciones para diversas configuraciones de topología y tamaño de la red, carga de tráfico, estructura de los conmutadores, tamaño de la memoria, y técnica de control de congestión empleada en la red. Los resultados de estos experimentos nos han permitido, en primer lugar, detectar y corregir ciertos errores del planteamiento inicial de *RECN*, y en segundo, confirmar la validez del mecanismo en su versión final, una vez mejorada la versión inicial con los cambios introducidos en el capítulo 4, y una vez ajustados los valores de sus parámetros críticos para un funcionamiento óptimo. Los resultados correspondientes a esta evaluación de la versión final de *RECN* demuestran que el mecanismo responde a lo que se esperaba de él, eliminando eficientemente el *HOL blocking* que pudiera aparecer en la red en cualquier circunstancia, y requiriendo siempre la misma cantidad moderada de recursos. Naturalmente, esta aportación implica cumplir los objetivos 4 y 5 planteados en la sección 1.4.

Como puede comprobarse, tras valorar las aportaciones de la tesis, todos los objetivos planteados para la misma (enumerados en la sección 1.4) pueden considerarse cumplidos.

6.2. Conclusiones

Aunque en algunos capítulos precedentes se han mostrado ya conclusiones parciales sobre los análisis incluidos en ellos, creemos oportuno el recordar, de forma resumida, las principales conclusiones que pueden extraerse de la tesis, y que son las siguientes:

- **La congestión es un serio problema para las actuales redes de interconexión de altas prestaciones**. Como se ha expuesto y demostrado a lo largo de esta memoria, las situaciones de congestión, debido al efecto del *HOL blocking*, pueden degradar drásticamente las prestaciones de la red. Más aún, teniendo en cuenta las restricciones debidas al coste de los componentes y al consumo de potencia, no es posible hoy en día sobredimensionar la red, lo que llevará a trabajar cerca del punto de saturación de la misma y, por tanto, a una mayor probabilidad de situaciones de congestión. Por consiguiente, es necesario emplear técnicas específicas de control de congestión para mantener al máximo las prestaciones de la red incluso en estas situaciones de congestión.

- **Los árboles de congestión pueden aparecer, expandirse y desaparecer de muy diversas formas**. Tal y como se ha mostrado en el estudio sobre la evolución de los árboles, las ideas tradicionales sobre el crecimiento y desaparición de los mismos no son ya válidas. Así, la congestión puede aparecer en entradas o salidas de los conmutadores, y expandirse en forma de árbol de congestión desde la raíz hacia las hojas, pero también en sentido contrario. Igualmente, un árbol de congestión puede colapsarse desde la raíz a las hojas, y viceversa. Esta variada evolución de los árboles se debe tanto a las diferentes arquitecturas de los conmutadores como a los muy diversos patrones de tráfico que pueden darse en la red. Por otra parte, estas características de los árboles de congestión deben ser tenidas en cuenta por cualquier técnica de control de congestión. De otro modo, como hemos visto, la técnica puede no ser eficaz en algunos casos, o bien no usar de forma eficiente los recursos destinados a controlar la congestión.

- **Las anteriores propuestas para el control de congestión ofrecen eficacia o escalabilidad, pero no ambas a la vez**. Muchas de las principales técnicas utilizadas anteriormente para solucionar los problemas asociados a la congestión presentan distintos problemas respecto a su eficacia: retardos excesivos entre la detección de la congestión y la aplicación de una solución (técnicas reactivas "clásicas"), eliminación incompleta del *HOL blocking* (*Virtual Output Queuing* a nivel de conmutador, *Destination Based Buffer Management*, *Virtual Channels* configurados como *VOQsw*), o falta de un enfoque específico para el control de congestión (técnicas proactivas, *Dynamically-Allocated Multi-Queues*, encaminamiento adaptativo). Estos problemas respecto a la eficacia se han puesto de manifiesto en los resultados ofrecidos en la presente memoria para algunas de estas técnicas, donde puede comprobarse claramente que su uso no evita la degradación de prestaciones en la red en situaciones de congestión. Por contra, otras técnicas ofrecen máxima eficacia, pero son difícilmente implementables en redes de medio o gran tamaño, debido a requerir demasiados recursos para eliminar eficazmente el *HOL blocking* (*Virtual Output Queuing* a nivel de red, *Virtual Channels* configurados como *VOQnet*). Este hecho resulta evidente en el estudio

presentado en esta memoria sobre el área de silicio requerida para implementar alguna de estas técnicas.

- ***RECN* elimina totalmente el *HOL blocking* producido por situaciones de congestión, empleando un número limitado de recursos**. Como se ha demostrado en los distintos análisis de prestaciones presentados en esta memoria, *RECN* es capaz de eliminar completamente el *HOL blocking* que pueda producirse en la red, manteniendo de este modo las prestaciones de la misma al máximo incluso en situaciones de congestión. En este sentido, *RECN* consigue unas prestaciones al mismo nivel que *VOQnet* (una técnica considerada "ideal") en cuanto a la productividad de la red, y mejora incluso los resultados conseguidos por esta última técnica en cuanto a latencia de los paquetes no congestionados. Además, tal y como se ha expuesto en la presente memoria, *RECN* consigue estos resultados con un número de recursos reducido e independiente del tamaño de la red (a diferencia de *VOQnet*), lo que implica a su vez, de cara a una posible implementación real, unos requisitos constantes y muy modestos en términos de área de silicio, como hemos visto. Por otra parte, también se ha demostrado que *RECN* elimina el *HOL blocking* de forma mucho más eficaz que otras técnicas con similares requisitos en cuanto a recursos (como *VOQsw*), que a diferencia de *RECN* no son capaces de mantener las prestaciones de la red a un buen nivel en caso de situaciones graves de congestión. Evidentemente, las ventajas de *RECN* sobre estas otras técnicas son el resultado de una gestión mucho más eficiente e inteligente de los recursos disponibles. En definitiva, *RECN* soluciona completamente el principal problema asociado a la congestión (el *HOL blocking*) de forma eficiente y escalable.

6.3. Trabajos publicados

Los distintos estudios, desarrollos y resultados recopilados en la presente memoria de tesis han dado lugar a varios artículos que han sido publicados (o están pendientes de serlo) en las actas de varios congresos y en revistas especializadas. A continuación se listan todas las publicaciones relacionadas con la presente tesis, indicando brevemente el contenido de cada una de ellas:

- J. Duato, I. Johnson, J. Flich, F. Naven, P.J. García, T. Nachiondo. *A New and Cost-Effective Congestion Management Strategy for Lossless MultiStage Interconnection Networks*, 11th Symposium on High-Performance Computer Architecture 2005 (HPCA-2005), Actas del congreso (ISBN 0-7695-2275-0), pp. 108-119, San Francisco, California (E.E.U.U.), Febrero de 2005.

En este artículo se plantean tanto la necesidad de contar con nuevas técnicas de control de congestión como las ideas básicas de la propuesta inicial de *RECN*. También se describe el funcionamiento general de los distintos aspectos del mecanismo (detección de congestión, propagación de la información de congestión, etc.) y se presenta una primera evaluación de sus prestaciones.

- P.J. García, J. Flich, J. Duato, I. Johnson, F. J. Quiles, F. Naven. *Dynamic Evolution of Congestion Trees: Analysis and Impact on Switch Architecture*, 1st International Conference on High Performance Embedded Architectures & Compilers (HiPEAC 2005), Lecture Notes in Computer Science 3793 (ISBN 3-540-30317-0, ISSN 0302-9743), Ed. Springer-Verlag, pp. 266-285, Barcelona, España, Noviembre de 2005.

En esta publicación se presenta un profundo estudio sobre la evolución de los árboles de congestión y los factores que la condicionan, analizando cómo algunos de los distintos tipos de crecimiento de árboles afectan negativamente al rendimiento de *RECN*. Además, se proponen los cambios necesarios para remediar estos problemas, que dan lugar a una versión mejorada de *RECN*, y se evalúan las ventajas de dicha versión sobre la anterior.

- P.J. García, J. Flich, J. Duato, F. J. Quiles, I. Johnson, F. Naven. *On the Correct Sizing on Meshes Through an Effective Congestion Management Strategy*, 11th International Euro-Par Conference, Lecture Notes in Computer Science 3648 (ISBN 3-540-28700-0, ISSN 0302-9743), Ed. Springer-Verlag, pp. 1035-1045, Lisboa, Portugal, Agosto de 2005.

- P.J. García, F.J. Quiles, J. Flich, J. Duato, I, Johnson, F. Naven. *Dimensionamiento eficiente en mallas mediante una técnica escalable de eliminación del HOL blocking*, XVI Jornadas de Paralelismo, Actas de las Jornadas, Ed. Thomson (ISBN 84-9732-430-7), pp. 149-156, Granada, España, Septiembre 2005.

En estos dos artículos se muestra cómo *RECN* aporta flexibilidad al dimensionamiento de la red, permitiendo configurar la misma con distintos números de conmutadores y de terminales por conmutador sin riesgo a que ésto afecte a las prestaciones obtenidas. En ambos casos *RECN* se aplica a topologías de malla, demostrándose que el mecanismo no es sólo válido para las redes multietapa, en las que se centraban los anteriores artículos.

- P.J. García, J. Flich, J. Duato, I. Johnson, F. J. Quiles, F. Naven. *RECN-DD: A Memory-Efficient Congestion Management Technique for Advanced Switching*, 35th International Conference on Parallel Processing (ICPP 2006), Actas del congreso (ISBN 0-7695-2636-5), pp. 23-32, Columbus, Ohio (E.E.U.U.), Agosto 2006.

En este trabajo se analiza cómo las diversas formas de desaparición de los árboles de congestión pueden llevar a que *RECN* use ineficientemente las *SAQs* disponibles, y se propone un nuevo mecanismo de liberación de las mismas que soluciona este problema. Además, se presentan resultados para evaluar el mecanismo mejorado y se estima el área de silicio requerida para una posible implementación del mismo.

- P.J. García, J. Flich, J. Duato, I. Johnson, F. J. Quiles, F.Naven. *Efficient, Scalable Congestion Management for Interconnection Networks*, IEEE Micro Magazine, Vol. 26, Nº 5 (ISSN 0272-1732), pp. 52-66, Septiembre/Octubre 2006.

En este artículo de revista se ofrece una amplia revisión de los distintos enfoques con lo que se ha abordado el control de congestión en redes de interconexión, analizando los motivos por los que ha resultado tradicionalmente complicado que las técnicas propuestas ofrecieran a la vez eficacia y escalabilidad. Igualmente, se plantean las claves para conseguir compatibilizar ambas, mostrando a *RECN* como ejemplo palpable de técnica eficiente y escalable, junto con resultados que confirman esta idea.

- A. Martínez, P.J. García, F.J. Alfaro, J. Flich, J.L. Sánchez, F.J. Quiles, J. Duato. *A Cost-Effective Interconnection Network Architecture with QoS and Congestion Management Support*, 12th International Euro-Par Conference, Lecture Notes in Computer Science 4128 (ISBN 3-540-37783-2, ISSN 0302-9743), Ed. Springer-Verlag, pp. 884-895, Dresden, Alemania, Agosto de 2006.

- A. Martínez, P.J. García, F.J. Alfaro, J.L. Sánchez, F.J. Quiles, J. Flich, J. Duato. *Una arquitectura eficiente de red de interconexión con soporte de calidad de servicio y control de congestión*, XVII Jornadas de Paralelismo, Actas de las Jornadas (ISBN 84-690-0551-0), pp. 187-192, Albacete, España, Septiembre de 2006.

Estos dos artículos no surgen directamente de la materia expuesta en la presente memoria, sino de una aplicación de *RECN* en entornos con requisitos de calidad de servicio, en combinación con otra técnica propuesta en este sentido que comparte con *RECN* la economía de recursos [MAS05]. Hemos considerado oportuno incluir ambas publicaciones en esta lista como muestra de las posibilidades de *RECN*, y de su impacto inmediato en el campo de las redes de interconexión.

6.4. Financiación disfrutada

La presente tesis se ha desarrollado en el marco de varios proyectos de investigación financiados por diferentes instituciones. A continuación se indican los datos más relevantes de dichos proyectos:

- **Título del proyecto:** Mejora de las prestaciones y servicios ofrecidos por las redes de computadores personales. Desarrollo de aplicaciones multimedia distribuidas.

  Entidad financiadora: C.I.C.Y.T (TIC2000-1151-C07-02)

  Entidades participantes: U. de Castilla-La Mancha, U. Politécnica de Valencia, U. de Murcia, U. de Valencia, U. Jaime I

  Duración, desde: Enero de 2000 **hasta:** Diciembre de 2003

  Investigador principal: Antonio Garrido del Solo

  Número de investigadores participantes: 9

  Importe: 25.200.000 ptas

- **Título del proyecto:** Redes y Arquitecturas de Altas Prestaciones.

  Entidad financiadora: Convocatoria Grupos Consolidados, JCCM, GC-020-017

  Entidades participantes: Universidad de Castilla-La Mancha

  Duración, desde: Enero de 2002 **hasta:** Diciembre de 2004

  Investigador principal: Francisco J. Quiles Flor

  Número de investigadores participantes: 15

  Importe: 60.000,00 euros

- **Título del proyecto:** Diseño de estrategias avanzadas para calidad de servicio y reconfiguración en redes Infiniband.

  Entidad financiadora: Consejería de Ciencia y Tecnología de la Junta de Comunidades de Castilla-La Mancha, PBC-02-008

  Entidades participantes: Universidad de Castilla-La Mancha

  Duración, desde: Julio de 2002 **hasta:** Julio de 2005

  Investigador principal: Jose Luis Sánchez García

  Número de investigadores participantes: 8

  Importe: 108.241,00 euros

- **Título del proyecto:** Mejora de las prestaciones de arquitecturas de redes SAN/LSAN para la difusión de tráfico multimedia con soporte de QoS.

  Entidad financiadora: C.I.C.Y.T. (TIC2003-08154-C06-02)

  Entidades participantes: U. Castilla-La Mancha, U. Politécnica de Valencia, U. Murcia, U. de Valencia, U. Jaume I

  Duración, desde: Septiembre de 2003 **hasta:** Septiembre de 2006

  Investigador principal: Francisco J. Quiles Flor

  Número de investigadores participantes: 14

  Importe: 226.400,00 euros

- **Título del proyecto:** Técnicas eficientes de calidad de servicio y control de congestión para la mejora de las prestaciones de los servidores de Internet.

  Entidad financiadora: Junta de Comunidades de Castilla-La Mancha (PBC-05-005)

  Entidades participantes: Universidad de Castilla-La Mancha, Universidad Politécnica de Valencia

  Duración, desde: Enero de 2005 **hasta:** Diciembre de 2007

  Investigador principal: Jose Luis Sánchez García

  Número de investigadores participantes: 12

  Importe: 56.000,00 euros

- **Título del proyecto:** Mejora de la infraestructura dedicada a proporcionar servicios de Internet.

  Entidad financiadora: Fondo Social Europeo y Junta de Comunidades de Castilla-La Mancha(PREG-05-25)

  Entidades participantes: Universidad de Castilla-La Mancha

  Duración, desde: Diciembre de 2005 **hasta:** Noviembre de 2007

  Investigador principal: Jose Luis Sánchez García

  Número de investigadores participantes: 7

  Importe: 80.280,00 euros

6.5. Trabajo futuro

La línea de investigación reflejada en la presente memoria de tesis podría servir de base para la realización de otros estudios. En este sentido, a continuación se indican algunas ideas que podrían desarrollarse en un futuro más o menos próximo:

- **Implementación de *RECN* en sistemas reales**. Aunque la evaluación mediante simulación del mecanismo final detallada en esta memoria se ha realizado con el mayor cuidado y la máxima precisión posibles, es evidente que una implementación real demostraría sin lugar a dudas la valía de *RECN* como técnica de control de congestión eficiente y escalable. En este sentido, se han dado ya pasos significativos para incluir *RECN* en el estándar *Advanced Switching*, lo que abre totalmente las puertas a que los distintos fabricantes de componentes *AS* incorporen *RECN* a sus productos. Tampoco descartamos una implementación "propia" de *RECN* mediante *FPGAs*.

- **Modificación de algunos aspectos del mecanismo para facilitar aún más su implementación**. Aunque en todo momento *RECN* se ha diseñado desde una perspectiva realista, y siempre con la idea de que el mecanismo fuera aplicado en la práctica, existen ciertos aspectos que podrían suponer una cierta complicación de cara a una implementación real. A continuación se indican estos aspectos del mecanismo que sería interesante modificar:

  - Sistema de detección de congestión en las entradas. Las colas de detección, empleadas para detectar salidas congestionadas desde las entradas de los conmutadores, son una solución correcta pero probablemente no "ideal", ya que su existencia implica que exista una diferente organización de memoria en entradas y salidas, y además aumentan (aunque muy ligeramente) los

requisitos mínimos de área de silicio en las entradas. Si fuese posible encontrar un método distinto para detectar, desde las entradas, qué salida es la "responsable" de la congestión, se evitarían estas molestias. En este sentido, existen varias posibilidades que pueden evaluarse: detectar la salida congestionada a partir de la solicitada por el último paquete almacenado, o por aquél que está en cabeza de la cola, etc..

- Procesado de la información de encaminamiento de los paquetes. Tal y como se ha explicado exhaustivamente, *RECN* requiere que la información de encaminamiento de los paquetes se compare con la almacenada en los distintos registros de la *CAM* del puerto antes de decidir en qué cola se almacena dicho paquete. Esto implica que es necesario una pequeña zona de memoria "de tránsito" (*landing pad*) donde almacenar el paquete hasta que se determine en qué cola debe almacenarse definitivamente. Para evitar esto, una posibilidad que podría evaluarse es procesar la información de encaminamiento de cada paquete "a posteriori", tras almacenar el paquete en la cola estándar, y enviarlo a otra cola del mismo puerto, si es necesario, cuando llegue a la cabeza de dicha cola. Nótese que, al poder "desviar" los paquetes congestionados a *SAQs*, no se produciría *HOL blocking* aun habiendo sido éstos almacenados en la cola estándar.

- **Combinación de *RECN* con técnicas de inhibición de inyección**. La propagación de la información de congestión que *RECN* lleva a cabo para comunicar la posición de la raíz de un árbol de congestión podría perfectamente llegar hasta las propias fuentes de dicha congestión (terminales de origen de los paquetes pertenecientes al árbol). Por tanto, puede plantearse el emplear esta información para inhibir la generación de paquetes dirigidos hacia la raíz del árbol desde dichas fuentes. Téngase en cuenta que, aunque esta filosofía corresponde a una técnica de control de congestión "reactiva" clásica, cuyos inconvenientes hemos analizado, en esta ocasión, se trataría de un control de congestión "secundario", añadido al ya realizado por *RECN*, por lo que dichos inconvenientes no existirían. Por consiguiente, es una posibilidad que quizá merezca la pena considerar.

- **Adaptación de *RECN* a redes con conmutación tipo *Wormhole***. Como se ha indicado en el planteamiento del mecanismo, *RECN* asume el uso de *Virtual Cut-Through* como técnica de conmutación. Algunos aspectos de *RECN* dependen directamente de este supuesto, y no serían adecuados si la red empleara conmutación de tipo *Wormhole*. Por ejemplo, la detección de congestión mediante la comprobación del nivel de ocupación de las colas no sería aplicable en los pequeños *buffers* característicos de los conmutadores que usan *Wormhole*. Sin embargo, creemos que, realizando las modificaciones necesarias, el planteamiento de *RECN* es perfectamente exportable a redes con conmutación tipo *Wormhole*. De hecho, ya hemos iniciado las primeras aproximaciones para hacer posible el uso

de *RECN* en la arquitectura de comunicación *EXTOLL*, que usa precisamente *Wormhole* como técnica de conmutación.

- **Aplicación de *RECN* a entornos con requisitos de calidad de servicio**. En este caso, más que de un "trabajo futuro" deberíamos hablar de un "trabajo presente", pues ya existen propuestas e incluso algunos resultados en este sentido (como ya hemos indicado en la lista de publicaciones relacionadas con la tesis). Sin embargo, pensamos que las posibilidades de *RECN* como soporte para la calidad de servicio son enormes, y aún pueden explotarse mucho más. Nótese que, a diferencia de otras técnicas de control de congestión, *RECN* sí identifica con exactitud qué paquetes forman parte de árboles de congestión, y esta información puede ser empleada por el árbitro del conmutador para conceder más o menos preferencia a estos u otros paquetes. La variedad de políticas basadas en esta información que podrían evaluarse es notable, máxime si se aplican en entornos con diferentes clases de tráfico, donde la congestión puede aparecer "selectivamente" en función de la clase a la que los paquetes pertenezcan.

Bibliografía

[ABC99] F.J. Alfaro, A. Bermúdez, R .Casado, P.J. García, J. Duato, F.J. Quiles, J.L.Sánchez. *Reconfiguración dinámica en NOWs: un paso hacia la garantía de QoS.* In Proc. X Jornadas de Paralelismo, Sep. 1999.

[Aea92] A. Asthana, et al. *Toward a gigabit ip router.* High Speed Networks, Vol. 1, No. 4, pp. 281–288, 1992.

[AOS93] T. Anderson, S. Owicki, J. Saxe, C. Thacker. *High-speed switch scheduling for local-area networks.* ACM Transactions on Computer Systems, Vol. 11, No. 4, pp. 319–352, November 1993.

[ASD04] F.J. Alfaro, J.L. Sánchez, J. Duato. *Qos in infiniband subnetworks.* IEEE Transactions on Parallel and Distributed Systems, Vol. 15, No. 9, 2004.

[ASIa] ASI-SIG. *Advanced Switching Core Architecture Specification.* http://www.asi-sig.org/specifications.

[ASIb] ASI-SIG. *Advanced Switching for the PCI Express Architecture.* http://www.intel.com/technology/pciexpress/devnet/AdvancedSwitching.pdf.

[Bak95] F. Baker. Rfc 1812: Requirements for IP version 4 routers, june 1995.

[BB00] T. Burd, R. Brodensen. *Design issues for dynamic voltage scaling.* In Proc. of the Int. Symposium on Low Power Electronics and Design, pp. 9–14, July 2000.

[BCF95] N.J. Boden, D. Cohen, R.E. Felderman, et al. *Myrinet - a gigabit per second local area network.* IEEE Micro, pp. 29–36, February 1995.

[BDH03] L.A. Barroso, J. Dean, U. Holzle. *Web search for a planet: the google cluster architecture.* IEEE Micro, pp. 22–28, March-April 2003.

[BL03] E. Baydal, P. López. *A robust mechanism for congestion control: INC.* In Proc. 9th International Euro-Par Conference, pp. 958–968, August 2003.

[BLD00] E. Baydal, P. López, J. Duato. *A simple and efficient mechanism to prevent saturation in wormhole networks.* In Proc. of the Int. Parallel and Distributed Processing Symposium, May 2000.

[BLD01] E. Baydal, P. López, J. Duato. *A congestion control mechanism for wormhole networks.* In Proc. of 9th Euromicro Workshop on Parallel and Distributed Processing, pp. 19–26, February 2001.

[BLD02] E. Baydal, P. López, J. Duato. *Avoiding network congestion with local information.* In Proc. Int. Symposium on High Performance Computing, May 2002.

[BLK93] R. Bianchini, T. J. LeBlanc, L. I. Kontothanassis, M. E. Crovella. *Alleviating memory contention in matrix computations on large-scale shared-memory multiprocessors.* Technical report, Dept. of Computer Science, Rochester University, April 1993. Tech. Report 449.

[Com] Internet Request For Comment. Rfcs 793,761,675: Transmission control protocol (tcp).

[DA93] W. J. Dally, H. Aoki. *Deadlock-free adaptive routing in multicomputer networks using virtual channels.* IEEE Transactions on Parallel and Distributed Systems, Vol. 4, No. 4, pp. 466–475, April 1993.

[Dal92] W. J. Dally. *Virtual-channel flow control.* IEEE Transactions on Parallel and Distributed Systems, Vol. 3, No. 2, pp. 194–205, March 1992.

[Dan99] S. P. Dandamudi. *Reducing hot-spot contention in shared-memory multiprocessor systems.* IEEE Concurrency, Vol. 7, No. 1, pp. 48–59, January 1999.

[DB95] J. Ding, L. Bhuyan. *Evaluation of multi-queue buffered multistage interconnection networks under uniform and nonuniform traffic patterns.* In Proc. 4th Int. Conf. on Computer Communication and Networks, pp. 576–583, 1995.

[DCD98] W. J. Dally, P. Carvey, L. Dennison. *The avici terabit switch/router.* In Proc. 6th Hot Interconnects, August 1998.

[DFN04] J. Duato, J. Flich, T. Nachiondo. *Cost-effective technique to reduce hol blocking in single-stage and multistage switch fabrics.* In Proc of Euromicro 2004 Conference on Parallel, Distributed and Network-based Processing, pp. 48–53, February 2004.

[dP95] M. de Prycker. *Asynchronous Transfer Mode: Solution for Broadband ISDN.* Ed. Prentice Hall, 1995.

[DS86] W. J. Dally, C. L. Seitz. *The torus routing chip.* Journal of Distributed Computing, Vol. 1, No. 3, pp. 187–196, October 1986.

[DS87] W. J. Dally, C. L. Seitz. *Deadlock-free message routing in multiprocessor interconnection networks.* IEEE Transactions on Computers, Vol. C–36, No. 5, pp. 547–553, May 1987.

[Dua93] J. Duato. *A new theory of deadlock-free adaptive routing in wormhole networks.* IEEE Transactions on Parallel and Distributed Systems, Vol. 4, No. 12, pp. 1320–1331, December 1993.

[DYN02] J. Duato, S. Yalamanchili, L. Ni. *Interconnection networks. An engineering approach.* Morgan Kaufmann Publishers, 2002.

[EM00] A. Engbersen, C. Minkenberg. *A combined input and output queued packet-switched system based on a prizma switch-on-a-chip technology.* IEEE Communication Magazine, Vol. 38, No. 12, pp. 70–77, December 2000.

[FGL99] D. Franco, I. Garces, E. Luque. *A new method to make communication latency uniform: Distributed routing balancing.* In ACM International Conference on Supercomputing, May 1999.

[Fli01] J. Flich. *Mejora de las prestaciones de la redes de estaciones de trabajo con encaminamiento fuente.* PhD thesis, Departamento de Informática de Sistemas y Computadores de la Universidad Politécnica de Valencia, 2001.

[GMA02] P.J. García, M.D. Mora, F.J. Alfaro, J.L. Sánchez, J. Flich. *Evaluation of alternative arbitration policies for myrinet switches.* In Proceedings of Workshop on Communication Architecture for Clusters, April 2002.

[Gro] Advanced Switching Interconnects Special Interest Group. Página web de ASI-SIG. http://www.asi-sig.org/home.

[GY93] P. T. Gaughan, S. Yalamanchili. *Adaptive routing protocols for hypercube interconnection networks.* IEEE Computer, Vol. 26, No. 5, May 1993.

[HA97] C. Hyatt, D. P. Agrawal. *Congestion control in the wormhole-routed torus with clustering and delayed deflection.* In Proc. Workshop on Parallel Computing, Routing, and Communication, June 1997.

[HP03] J.L. Hennessy, D.A. Patterson. *Computer Architecture: A Quantitative Approach (3^a edición).* Morgan Kaufmann Publishers, 2003.

[IBA] http://www.infinibandta.org/specs.

[Inc] Myricom Inc. Página web de Myricom. http://www.myri.com/.

[Jai91] R. Jain. *The art of computer system performance analysis: techniques for experimental design, measurement, simulation and modeling.* John Wiley and Sons, Inc., 1991.

[KDT05] J. Kim, W. J. Dally, B. Towles, A. K. Gupta. *Microarchitecture of a high.-radix router.* In Proc. 32nd Annual International Symposium on Computer Architecture, June 2005.

[KHM87] M. Karol, M. Hluchyj, S. Morgen. *Input versus output queueing on a space division switch.* IEEE Transactions on Communications, Vol. 35, No. 12, pp. 1347–1356, 1987.

[KK79] P. Kermani, L. Kleinrock. *Virtual cut-through: A new computer communication switching technique.* Computer Networks, Vol. 3, pp. 267–286, 1979.

[KLC94] J. H. Kim, Z. Liu, A. A. Chien. *Compressionless routing: A framework for adaptive and fault-tolerant routing.* In Proc. of the 21st Int. Symposium on Computer Architecture, April 1994.

[KLC97] J. H. Kim, Z. Liu, A. A. Chien. *Compressionless routing: A framework for adaptive and fault-tolerant routing.* IEEE Trans. on Parallel and Distributed Systems, Vol. 8, No. 3, 1997.

[KM04] V. Krishnan, D. Mayhew. *A Localized Congestion Control Mechanism for PCI Express Advanced Switching Fabrics.* In Proc. 12th IEEE Symp. on Hot Interconnects, August 2004.

[KS91] S. Konstantinidou, L. Snyder. *Chaos router: Architecture and performance.* In Proc. 18th International Symposium on Computer Architecture, pp. 79–88, June 1991.

[KSS98] M. Katevenis, D. Serpanos, E. Spyridakis. *Credit-flow-controlled atm for mp interconnection: the atlas i single-chip atm switch.* In Proc. of the 4th Int. Symp. on High-Performance Computer Architecture, pp. 47–56, February 1998.

[Lab] Sandia Laboratories. Página web de Sandia National Laboratories. http://www.sandia.gov.

[lc] MCR linux cluster. Página web del MCR linux cluster. http://www.llnl.gov/mcr/.

[LD93] P. López, J. Duato. *Deadlock-free adaptive routing algorithms for the 3d-torus: limitations and solutions.* In Proc. Parallel Architectures Languages Europe, pp. 684–687, June 1993.

[lis] Top 500 list. Página web de la lista Top 500. http://www.top500.org.

[LMD97] P. López, J. M. Martínez, J. Duato, F. Petrini. *On the reduction of deadlock frequency by limiting message injection in wormhole networks.* In Proc. of the 1997 Parallel Computer Routing and Communication Workshop, pp. 295–307, 1997.

[LMD98] P. López, J. M. Martínez, J. Duato. *Dril: Dynamically reduced message injection limitation mechanism for wormhole networks.* In Proc. of the 1998 Int. Conference on Parallel Processing, pp. 535–542, 1998.

[LSC95] J. Liu, K. G. Shin, C. C. Chang. *Prevention of congestion in packet-switched multistage interconnection networks.* IEEE Transactions on Parallel and Distributed Systems, Vol. 6, No. 5, pp. 535–541, May 1995.

[Ltd] Xyratex Ltd. Página web de Xyratex. http://www.xyratex.com/.

[MAG03] C. Minkenberg, F. Abel, M. Gusat, R. P. Luijten, W. Denzel. *Current issues in packet switch design.* In ACM SIGCOMM Computer Communication Review, January 2003.

[Mar] Supercomputador MareNostrum. Página web del supercomputador MareNostrum. http://www.bsc.es/resources/marenostrum,es.htm.

[MAS05] A. Martínez, F. J. Alfaro, J. L. Sánchez, J. Duato. *Providing full QoS support in clusters using only two VCs at the switches.* Lecture Notes in Computer Science (HiPC 2005), Vol. 3769, pp. 158–169, December 2005.

[MBL02] S.S. Mukherjee, P. Bannon, S. Lang, A. Spink, D. Webb. *The alpha 21364 network architecture.* IEEE Micro, Vol. 22, No. 1, pp. 26–35, January-February 2002.

[Muk97] B. Mukherjee. *Optical Communication Networks.* Mc-Graw-Hill, 1997.

[Pea98] C. Partridge, et al. *A 50-gb/s ip router.* IEEE/ACM Trans. Networking, Vol. 6, pp. 237–248, June 1998.

[Pea99] K. Pattabhiraman, et al. *On the speedup required for work-conserving crossbar switches.* IEEE Journal on Selected Areas in Communications, Vol. 17, No. 6, pp. 1057–1066, June 1999.

[PN85] G. Pfister, A. Norton. *Hot spot contention and combining in multistage interconnect networks.* IEEE Transactions on Computers, Vol. C-34, pp. 943–948, October 1985.

[PV96] F. Petrini, M. Vanneschi. *Minimal adaptive routing with limited injection on toroidal k-ary n-cubes.* In Proc. of Supercomputing, 1996.

[QsN] Quadrics QsNet. Página web de Quadrics. http://doc.quadrics.com.

[RCOC01] R. Rojas-Cessa, E. Oki, H. Chao. *Cixob-k: Combined input-crosspoint-output buffered packet switch.* In Proc. of the IEEE Global Telecomunications Conference, 2001.

[Rea99] R. Riesen, et al. *Cplant.* In Proc. of the Second Extreme Linux Workshop, June 1999.

[RS98] R. Ramaswami, K. Sivarajan. *Optical Networks: A Practical Perspective.* Morgan Kaufmann, 1998.

[San02] J.C. Sancho. *Contribución al diseño de algoritmos de encaminamiento en redes de estaciones de trabajo.* PhD thesis, Departamento de Informática de Sistemas y Computadores de la Universidad Politécnica de Valencia, 2002.

[SC] Earth Simulator, T.E.S. Center. Página web del Earth Simulator Center. http://www.es.jamstec.go.jp/esc/eng/index.html.

[SC04] J. M. Stine, N. P. Carter. *Comparing adaptive routing and dynamic voltage scaling for link power reduction.* Computer Architecture Letters, Vol. 3, No. 1, pp. 14–17, July 2004.

[SDT04] A. Singh, W. J. Dally, B. Towles, A. K. Gupta. *Globally adaptive load-balanced routing on tori.* Computer Architecture Letters, Vol. 3, No. 1, pp. 6–9, July 2004.

[Sea91] M. D. Schroeder, et all. *Autonet: A high-speed, self-configuring local area network using point-to-point links.* IEEE Journal on Selected Areas in Communications, Vol. 9, No. 8, pp. 1318–1334, October 1991.

[SG94] S. L. Scott, J. R. Goodman. *The impact of pipelined channels on k-ary n-cube networks.* IEEE Transactions on Parallel and Distributed Systems, Vol. 5, No. 1, pp. 2–16, January 1994.

[SLS90] G. S. Sohi S. L. Scott. *The use of feedback in multiprocessors and its application to tree saturation control.* IEEE Transactions on Parallel and Distributed Systems, Vol. 1, No. 4, pp. 385–398, October 1990.

[Spe] PCI Express Specifications. Página web de PCI-Express. http://www.pcisig.com/specifications/pciexpress/.

[SPJ03] L. Shang, L. S. Peh, N. K. Jha. *Dynamic voltage scaling with links for power optimization of interconnection networks.* In Proc. of 9th. Int. Symp. on High Performance Computer Architecture, pp. 91–102, February 2003.

[ssp] SSP homepage. `http://ginger.hpl.hp.com/research/itc/csl/ssp/`.

[ST98] A. Smai, L. Thorelli. *Global reactive congestion control in multicomputer networks.* In Proc. 5th Int. Conference on High Performance Computing, 1998.

[Sup] ASCI Red TOPS Supercomputer. Página web del supercomputador ASCI Red. http://www.sandia.gov/ASCI/Red/.

[Tan96] A.S. Tanenbaum. *Computer Networks (3ª edición).* Prentice Hall, 1996.

[Tea02] IBM BlueGene/L Team. *An overview of bluegene/l supercomputer.* In ACM Supercomputing Conference, 2002.

[TF88] Y. Tamir, G.L. Frazier. *High-performance multi-queue buffers for vlsi communication switches.* In Proc. 15th Int. Symposium on Computer Architecture, June 1988.

[TF92] Y. Tamir, G.L. Frazier. *Dynamically-allocated multi-queue buffers for vlsi communication switches.* IEEE Transactions on Computers, Vol. 41, No. 6, June 1992.

[TLM01] M. Thottetodi, A.R. Lebeck, S.S. Mukherjee. *Self-tuned congestion control for multiprocessor networks.* In Proc. of 7th. Int. Symp. on High Performance Computer Architecture, February 2001.

[TLM03] M. Thottetodi, A.R. Lebeck, S.S. Mukherjee. *Blam: A high-performance routing algorithm for virtual cut-through networks.* In Proc. of Int. Parallel and Distributed Processing Symp., April 2003.

[Vea00] W. Vogels, et al. *Tree-saturation control in the ac3 velocity cluster interconnect.* In Proc. 8th Conference on Hot Interconnects, August 2000.

[WSN95] M. Wang, H. J. Siegel, M. A. Nichols, S. Abraham. *Using a multipath network for reducing the effects of hot spots.* IEEE Transactions on Parallel and Distributed Systems, Vol. 6, No. 3, pp. 252–268, March 1995.

[YR95] C. Q. Yang, A. V. S. Reddy. *A taxonomy for congestion control algorithms in packet switching networks.* IEEE Network, Vol. 9, No. 5, pp. 34–45, July/August 1995.

[YTL87] P. Yew, N. Tzeng, D. H. Lawrie. *Distributing hot-spot addressing in large-scale multiprocessors.* IEEE Transactions on Computers, Vol. 36, No. 4, pp. 388–395, April 1987.